

MouseEventHandler delegate (System.Windows.Forms)

a) *ToString*

Description

Represents the method that will handle the **MouseDown** , **MouseUp** , or **MouseMove** event of a form, control, or other component. The source of the event. A **System.Windows.Forms.MouseEventArgs** that contains the event data.

When you create a **System.Windows.Forms.MouseEventHandler** delegate, you identify the method that will handle the event. To associate the event with your event handler, add an instance of the delegate to the event. The event handler is called whenever the event occurs, unless you remove the delegate. For more information about handling events with delegates, see .

NativeWindow class (System.Windows.Forms)

a) *ToString*

Description

Provides a low-level encapsulation of a window handle and a window procedure.

This class automatically manages window class creation and registration.

b) *NativeWindow*

Example Syntax:

c) *ToString*

[C#]	public	NativeWindow();
[C++]	public:	NativeWindow();

[VB]	Public	Sub	New()
------	--------	-----	-------

[JScript] public function NativeWindow();

d) *Handle*

e) *ToString*

[C#]	public	IntPtr	Handle	{get;}
------	--------	--------	--------	--------

[C++]	public:	__property	IntPtr	get_Handle();
-------	---------	------------	--------	---------------

[VB]	Public	ReadOnly	Property	Handle	As	IntPtr
------	--------	----------	----------	--------	----	--------

[JScript]	public	function	get	Handle()	:	IntPtr;
-----------	--------	----------	-----	----------	---	---------

Description

Gets the handle for this window.

Use this method when calling Windows API methods that require a handle for a window or control.

f) *AssignHandle*

[C#]	public	void	AssignHandle(IntPtr	handle);
------	--------	------	---------------------	----------

[C++]	public:	void	AssignHandle(IntPtr	handle);
-------	---------	------	---------------------	----------

[VB]	Public	Sub	AssignHandle(ByVal	handle	As	IntPtr)
------	--------	-----	--------------------	--------	----	---------

[JScript]	public	function	AssignHandle(handle	:	IntPtr);
-----------	--------	----------	---------------------	---	----------

Description

Assigns a handle to this window.

System.Windows.Forms.NativeWindow.WndProc(System.Windows.Forms.Message@) intercepts window messages sent to the *handle* parameter. Use **System.Windows.Forms.NativeWindow.ReleaseHandle** to reset the

handle's window procedure to the default window procedure. The handle to assign to this window.

g) *CreateHandle*

```
[C#]      public      virtual      void      CreateHandle(CreateParams      cp);  
[C++]      public:      virtual      void      CreateHandle(CreateParams*      cp);  
[VB]      Overridable Public Sub CreateHandle(ByVal cp As CreateParams)  
[JScript]      public      function      CreateHandle(cp      :      CreateParams);
```

Description

Creates a window handle for this window.

The *cp* parameter specifies the values that are passed to the **NativeMethods.Windows.CreateWindowEx** API method to create a window and its handle. A **System.Windows.Forms.CreateParams** that specifies the creation parameters for this window.

h) *DefWndProc*

```
[C#]      public      void      DefWndProc(ref      Message      m);  
[C++]      public:      void      DefWndProc(Message*      m);  
[VB]      Public      Sub      DefWndProc(ByRef      m      As      Message)  
[JScript]      public      function      DefWndProc(m      :      Message);
```

Description

Invokes the default window procedure associated with this window. It is an error to call this method when the **System.Windows.Forms.NativeWindow.Handle** property is zero. The message that is currently being processed.

i) *DestroyHandle*

[C#]	public	virtual	void	DestroyHandle();
[C++]	public:	virtual	void	DestroyHandle();
[VB]	Overridable	Public	Sub	DestroyHandle()
[JScript]	public	function		DestroyHandle();

Description

Destroys the handle associated with this window.

This method sets the **System.Windows.Forms.NativeWindow.Handle** property to zero, and calls the **System.Windows.Forms.NativeWindow.OnHandleChange** method to reflect the change.

j) *Finalize*

[C#]				~NativeWindow();
[C++]				~NativeWindow();
[VB]	Overrides	Protected	Sub	Finalize()
[JScript]	protected	override	function	Finalize();

Description

Releases the resources associated with this window.

k) *FromHandle*

[C#]	public	static	NativeWindow	FromHandle(IntPtr	handle);
[C++]	public:	static	NativeWindow*	FromHandle(IntPtr	handle);

[VB] Public Shared Function FromHandle(ByVal handle As IntPtr) As
NativeWindow

[JScript] public static function FromHandle(handle : IntPtr) : NativeWindow;

Description

Retrieves the window associated with the specified handle.

Return Value: The **System.Windows.Forms.NativeWindow** associated with the specified handle. This method returns **null** when the handle does not have an associated window.

If you receive a handle from another method, use this method to retrieve the window associated with the handle. A handle to a window.

l) OnHandleChange

[C#] protected virtual void OnHandleChange();

[C++] protected: virtual void OnHandleChange();

[VB] Overridable Protected Sub OnHandleChange()

[JScript] protected function OnHandleChange();

Description

Specifies a notification method that is called when the handle for a window is changed.

This method is invoked when the value of the **System.Windows.Forms.NativeWindow.Handle** property has changed.

m) OnThreadException

[C#] protected virtual void OnThreadException(Exception e);

[C++] protected: virtual void OnThreadException(Exception* e);

[VB] Overridable Protected Sub OnThreadException(ByVal e As Exception)

[JScript] protected function OnThreadException(e : Exception);

Description

When overridden in a derived class, manages an unhandled thread exception.

Typically,

System.Windows.Forms.Application.OnThreadException(System.Exception) handles thread exceptions. An **System.Exception** that specifies the unhandled thread exception.

n) ReleaseHandle

[C#] public virtual void ReleaseHandle();

[C++] public: virtual void ReleaseHandle();

[VB] Overridable Public Sub ReleaseHandle()

[JScript] public function ReleaseHandle();

Description

Releases the handle associated with this window.

This method does not destroy the window handle. Instead, it sets the handle's window procedure to the default window procedure. It sets the **System.Windows.Forms.NativeWindow.Handle** property to zero and calls **System.Windows.Forms.NativeWindow.OnHandleChange** to reflect the change.

o) WndProc

[C#] protected virtual void WndProc(ref Message m);

[C++] protected: virtual void WndProc(Message* m);

[VB] Overridable Protected Sub WndProc(ByRef m As Message)

1 [JScript] protected function WndProc(m : Message);

3 *Description*

4 Invokes the default window procedure associated with this window.

5 This method is called when a window message is sent to the handle of the
6 window. A **System.Windows.Forms.Message** that is associated with the
current Windows message.

7 NavigateEventArgs class (System.Windows.Forms)

8 a) *WndProc*

11 *Description*

12 Provides data for the **System.Windows.Forms.DataGrid.Navigate** event.

13 For more information about handling events, see .

14 b) *NavigateEventArgs*

15 *Example Syntax:*

16 c) *WndProc*

18 [C#] public NavigateEventArgs(bool isForward);

19 [C++] public: NavigateEventArgs(bool isForward);

20 [VB] Public Sub New(ByVal isForward As Boolean)

21 [JScript] public function NavigateEventArgs(isForward : Boolean);

23 *Description*

24 Initializes a new instance of the
25 **System.Windows.Forms.NavigateEventArgs** class.

The *isForward* parameter value is assigned to the **System.Windows.Forms.NavigateEventArgs.Forward** property. **true** to navigate in a forward direction; otherwise, **false**.

d) *Forward*

e) *WndProc*

[C#]	public	bool	Forward	{get;}
[C++]	public:	__property	bool	get_Forward();
[VB]	Public	ReadOnly	Property Forward	As Boolean
[JScript]	public	function	get Forward()	: Boolean;

Description

Gets a value indicating whether to navigate in a forward direction.

NavigateEventHandler delegate (System.Windows.Forms)

a) *ToString*

Description

Represents the method that will handle the **System.Windows.Forms.NavigateEventArgs** event of a **System.Windows.Forms.DataGrid**. The source of the event. A **System.Windows.Forms.NavigateEventArgs** that contains the event data.

When you create a **System.Windows.Forms.NavigateEventArgs** delegate, you identify the method that will handle the event. To associate the event with your event handler, add an instance of the delegate to the event. The event handler is called whenever the event occurs, unless you remove the delegate. For more information about event handler delegates, see .

NodeLabelEditEventArgs class (System.Windows.Forms)

a) *ToString*

Description

Provides data for the **System.Windows.Forms.TreeView.BeforeLabelEdit** and **System.Windows.Forms.TreeView.AfterLabelEdit** events.

The **System.Windows.Forms.TreeView.AfterLabelEdit** event occurs when the user finishes editing the text for a tree node. The **System.Windows.Forms.TreeView.BeforeLabelEdit** event occurs when the user begins editing the text for a tree node. A **System.Windows.Forms.NodeLabelEditEventArgs** object specifies the new text to associate with the tree node, the tree node that contains the label to edit, and whether the edit operation has been canceled.

b) *NodeLabelEditEventArgs*

Example Syntax:

c) *ToString*

[C#] public NodeLabelEditEventArgs(TreeNode node);
[C++] public: NodeLabelEditEventArgs(TreeNode* node);
[VB] Public Sub New(ByVal node As TreeNode)
[JScript] public function NodeLabelEditEventArgs(node : TreeNode); Initializes a new instance of the **System.Windows.Forms.NodeLabelEditEventArgs** class.

Description

Initializes a new instance of the **System.Windows.Forms.NodeLabelEditEventArgs** class for the specified **System.Windows.Forms.TreeNode**.

The **System.Windows.Forms.NodeLabelEventArgs.Node** property is assigned the *node* parameter value. The tree node containing the text to edit.

d) *NodeLabelEventArgs*

Example Syntax:

e) *ToString*

```
[C#] public NodeLabelEventArgs(TreeNode node, string label);
[C++] public: NodeLabelEventArgs(TreeNode* node, String* label);
[VB] Public Sub New(ByVal node As TreeNode, ByVal label As String)
[JScript] public function NodeLabelEventArgs(node : TreeNode, label :
String);
```

Description

Initializes a new instance of the **System.Windows.Forms.NodeLabelEventArgs** class for the specified **System.Windows.Forms.TreeNode** and the specified text with which to update the tree node label.

The **System.Windows.Forms.NodeLabelEventArgs.Node** property is assigned the *node* parameter value, and the **System.Windows.Forms.NodeLabelEventArgs.Label** property is assigned the *label* parameter value. The tree node containing the text to edit. The new text to associate with the tree node.

f) *CancelEdit*

g) *ToString*

```
[C#] public bool CancelEdit {get; set;}
[C++] public: __property bool get_CancelEdit();public: __property void
set_CancelEdit(bool);
```


1 [VB] Public Property CancelEdit As Boolean

2 [JScript] public function get CancelEdit() : Boolean;public function set

3 CancelEdit(Boolean);

4
5 *Description*

6 Gets or sets a value indicating whether the edit has been canceled.

7 *h) Label*

8 *i) ToString*

9
10 [C#] public string Label {get;}

11 [C++] public: __property String* get_Label();

12 [VB] Public ReadOnly Property Label As String

13 [JScript] public function get Label() : String;

14
15 *Description*

16 Gets the new text to associate with the tree node.

17 *j) Node*

18 *k) ToString*

19
20 [C#] public TreeNode Node {get;}

21 [C++] public: __property TreeNode* get_Node();

22 [VB] Public ReadOnly Property Node As TreeNode

23 [JScript] public function get Node() : TreeNode;

1
2 *Description*

3 Gets the tree node containing the text to edit.

4 NodeLabelEditEventHandler delegate (System.Windows.Forms)

5 *a) ToString*

6
7
8 *Description*

9 Represents the method that will handle the
10 **System.Windows.Forms.TreeView.BeforeLabelEdit** and
11 **System.Windows.Forms.TreeView.AfterLabelEdit** events of a
12 **System.Windows.Forms.TreeView** control. The source of the event. A
13 **System.Windows.Forms.NodeLabelEditEventArgs** that contains the event
14 data.

15 When you create a **System.Windows.Forms.NodeLabelEditEventArgs**
16 delegate, you identify the method that will handle the event. To associate the
17 event with your event handler, add an instance of the delegate to the event. The
18 event handler is called whenever the event occurs, unless you remove the
19 delegate. For more information about event handler delegates, see .

20 NotifyIcon class (System.Windows.Forms)

21 *a) ToString*

22
23
24
25 *Description*

26 Specifies a component that creates an icon in the Windows System Tray. This
27 class cannot be inherited.

28 Icons in the Windows System Tray are short cuts to processes that are running
29 in the background of a computer, such as a virus protection program or a
30 volume control. These processes do not come with their own user interfaces.

b) NotifyIcon

Example Syntax:

c) ToString

```
[C#] public NotifyIcon();
[C++] public: NotifyIcon();
[VB] Public Sub New()
[JScript] public function NotifyIcon(); Initializes a new instance of the
System.Windows.Forms.NotifyIcon class.
```

Description

Initializes a new instance of the **System.Windows.Forms.NotifyIcon** class.

When a new **System.Windows.Forms.NotifyIcon** is created, the **System.Windows.Forms.NotifyIcon.Visible** property is set to **false**. You must set the **System.Windows.Forms.NotifyIcon.Visible** property to **true** in order to use the **System.Windows.Forms.NotifyIcon** you created. This instance will exist until its container releases it to garbage collection.

d) NotifyIcon

Example Syntax:

e) ToString

```
[C#] public NotifyIcon(IContainer container);
[C++] public: NotifyIcon(IContainer* container);
[VB] Public Sub New(ByVal container As IContainer)
[JScript] public function NotifyIcon(container : IContainer);
```

Description

1 Initializes a new instance of the **System.Windows.Forms.NotifyIcon** class
with the specified container.

2 When a new **System.Windows.Forms.NotifyIcon** is created, the
3 **System.Windows.Forms.NotifyIcon.Visible** property is set to **false**. You
must set the **System.Windows.Forms.NotifyIcon.Visible** property to **true** in
4 order to use the **System.Windows.Forms.NotifyIcon** you created. This
instance will exist until its container releases it to garbage collection. An
5 **System.ComponentModel.IContainer** that represents the container for the
System.Windows.Forms.NotifyIcon control.

6 *f) Container*

7 *g) ContextMenu*

8 *h) ToString*

9
10
11
12 *Description*

13 Gets or sets the context menu for the tray icon.

14 The menu is shown when a right-mouse click is performed on an icon in the
system tray. Context menus also are known as pop-up menus.

15 *i) DesignMode*

16 *j) Events*

17 *k) Icon*

18 *l) ToString*

19
20
21
22 *Description*

23 Gets or sets the current icon.

1 *m) Site*

2 *n) Text*

3 *o) ToString*

4
5
6 *Description*

7 Gets or sets the ToolTip text displayed when the mouse hovers over a system
8 tray icon.

9 If the text is **null** , no ToolTip is displayed.

10 *p) Visible*

11 *q) ToString*

12 [C#] public bool Visible {get; set;}

13 [C++] public: __property bool get_Visible();public: __property void
14 set_Visible(bool);

15 [VB] Public Property Visible As Boolean

16 [JScript] public function get Visible() : Boolean;public function set
17 Visible(Boolean);

18
19 *Description*

20 Gets or sets a value indicating whether the icon is visible in the Windows System
21 Tray.

22 Since the default value is false, in order for the icon to show up in the system
23 tray, **System.Windows.Forms.NotifyIcon.Visible** must be set to **true** by the
24 user. By setting this property, the user is prompted to program the rest of the
25 functionality for the icon.

r) ToString

[C#]	public	event	EventHandler	Click;
[C++]	public:	__event	EventHandler*	Click;
[VB]	Public	Event	Click As	EventHandler

Description

Occurs when the user clicks the icon in the system tray.

For information about handling events, see .

s) ToString

Description

Occurs when the user double-clicks the icon in the system tray.

For information about handling events, see .

t) ToString

[C#]	public	event	MouseEventHandler	MouseDown;
[C++]	public:	__event	MouseEventHandler*	MouseDown;
[VB]	Public	Event	MouseDown As	MouseEventHandler

Description

Occurs when the user presses the mouse button while the pointer is over the icon in the system tray.

For information about handling events, see .

u) *ToString*

```
[C#]      public      event      MouseEventHandler      MouseMove;
[C++]     public:     __event    MouseEventHandler*      MouseMove;
[VB]      Public      Event      MouseMove      As      MouseEventHandler
```

Description

Occurs when the user moves the mouse while the pointer is over the icon in the system tray.

For information about handling events, see .

v) *ToString*

```
[C#]      public      event      MouseEventHandler      MouseUp;
[C++]     public:     __event    MouseEventHandler*      MouseUp;
[VB]      Public      Event      MouseUp      As      MouseEventHandler
```

Description

Occurs when the user releases the mouse button while the pointer is over the icon in the system tray.

For information about handling events, see .

w) *Dispose*

```
[C#]      protected      override      void      Dispose(bool      disposing);
[C++]     protected:      void      Dispose(bool      disposing);
[VB]      Overrides      Protected      Sub      Dispose(ByVal      disposing      As      Boolean)
[JScript] protected      override      function      Dispose(disposing      :      Boolean);
```

Description

Disposes of the resources (other than memory) used by the **System.Windows.Forms.NotifyIcon** .

Call **System.Windows.Forms.NotifyIcon.Dispose(System.Boolean)** when you are finished using the **System.Windows.Forms.NotifyIcon** . The **System.Windows.Forms.NotifyIcon.Dispose(System.Boolean)** method leaves the **System.Windows.Forms.NotifyIcon** in an unusable state. After calling **System.Windows.Forms.NotifyIcon.Dispose(System.Boolean)** , you must release all references to the **System.Windows.Forms.NotifyIcon** so the memory it was occupying can be reclaimed by garbage collection.

NumericUpDown class (System.Windows.Forms)

a) ToString

Description

Represents a Windows up-down control that displays numeric values.

A **System.Windows.Forms.NumericUpDown** control contains a single numeric value that can be incremented or decremented by clicking the up or down buttons of the control. The user may also enter in a value, unless the **System.Windows.Forms.NumericUpDown.ReadOnly** property is set to **true** .

b) NumericUpDown

Example Syntax:

c) ToString

[C#]	public	NumericUpDown();
[C++]	public:	NumericUpDown();
[VB]	Public	Sub New()
[JScript]	public	function NumericUpDown();

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25

Description

Initializes a new instance of the **System.Windows.Forms.NumericUpDown** class.

- d) *AccessibilityObject*
- e) *AccessibleDefaultActionDescription*
- f) *AccessibleDescription*
- g) *AccessibleName*
- h) *AccessibleRole*
- i) *ActiveControl*
- j) *AllowDrop*
- k) *Anchor*
- l) *AutoScroll*
- m) *AutoScrollMargin*
- n) *AutoScrollMinSize*
- o) *AutoScrollPosition*
- p) *BackColor*
- q) *BackgroundImage*
- r) *BindingContext*
- s) *BorderStyle*
- t) *Bottom*
- u) *Bounds*
- v) *CanFocus*
- w) *CanSelect*
- x) *Capture*
- y) *CausesValidation*
- z) *ChangingText*

- 1 *aa) ClientRectangle*
- 2 *bb) ClientSize*
- 3 *cc) CompanyName*
- 4 *dd) Container*
- 5 *ee) ContainsFocus*
- 6 *ff) ContextMenu*
- 7 *gg) Controls*
- 8 *hh) Created*
- 9 *ii) CreateParams*
- 10 *jj) Cursor*
- 11 *kk) DataBindings*
- 12 *ll) DecimalPlaces*
- 13 *mm) ToString*

17 *Description*

18 Gets or sets the number of decimal places to display in the up-down control.

19 When the **System.Windows.Forms.NumericUpDown.DecimalPlaces**
 20 property is set, the **System.Windows.Forms.NumericUpDown.UpdateEditText** method is
 21 called to update the up-down control's display to the new format.

1 **nn) DefaultImeMode**

2 **oo) DefaultSize**

3 **pp) DesignMode**

4 **qq) DisplayRectangle**

5 **rr) Disposing**

6 **ss) Dock**

7 **tt) DockPadding**

8 **uu) Enabled**

9 **vv) Events**

10 **ww) Focused**

11 **xx) Font**

12 **yy) FontHeight**

13 **zz) ForeColor**

14 **aaa) Handle**

15 **bbb) HasChildren**

16 **ccc) Height**

17 **ddd) Hexadecimal**

18 **eee) ToString**

19
20
21
22 *Description*

23 Gets or sets a value indicating whether the up-down control should display the
24 value it contains in hexadecimal format.

When the **System.Windows.Forms.NumericUpDown.Hexadecimal** property is set, the **System.Windows.Forms.NumericUpDown.UpdateEditText** method is called to update the up-down control's display to the new format.

fff) HScroll

ggg) ImeMode

hhh) Increment

iii) ToString

Description

Gets or sets the value to increment or decrement the up-down control when the up or down buttons are clicked.

Clicking the up button causes the **System.Windows.Forms.NumericUpDown.Value** property to increment by the amount specified by the **System.Windows.Forms.NumericUpDown.Increment** property and approach the **System.Windows.Forms.NumericUpDown.Maximum** property. Clicking the down button causes the **System.Windows.Forms.NumericUpDown.Value** property to be decremented by the amount specified by the **System.Windows.Forms.NumericUpDown.Increment** property and approach the **System.Windows.Forms.NumericUpDown.Minimum** property.

1 *jjj) InterceptArrowKeys*

2 *kkk) InvokeRequired*

3 *lll) IsAccessible*

4 *mmm) IsDisposed*

5 *nnn) IsHandleCreated*

6 *ooo) Left*

7 *ppp) Location*

8 *qqq) Maximum*

9 *rrr) ToString*

10
11
12 *Description*

13 Gets or sets the maximum value for the up-down control.

14 When the **System.Windows.Forms.NumericUpDown.Maximum** property is
15 set, the **System.Windows.Forms.NumericUpDown.Minimum** property is
16 evaluated and the

17 **System.Windows.Forms.NumericUpDown.UpdateEditText** method is
18 called. If the **System.Windows.Forms.NumericUpDown.Minimum** property
19 is greater than the new

20 **System.Windows.Forms.NumericUpDown.Maximum** property, the
21 **System.Windows.Forms.NumericUpDown.Minimum** property value is set
22 equal to the **System.Windows.Forms.NumericUpDown.Maximum** value. If
23 the current **System.Windows.Forms.NumericUpDown.Value** is greater than
24 the new **System.Windows.Forms.NumericUpDown.Maximum** value, the
25 **System.Windows.Forms.NumericUpDown.Value** property value is set equal
26 to the **System.Windows.Forms.NumericUpDown.Maximum** value.

sss) *Minimum*

ttt) *ToString*

[C#] public decimal Minimum {get; set;}

[C++] public: __property Decimal get_Minimum();public: __property void
set_Minimum(Decimal);

[VB] Public Property Minimum As Decimal

[JScript] public function get Minimum() : Decimal;public function set
Minimum(Decimal);

Description

Gets or sets the minimum allowed value for the up-down control.

When the **System.Windows.Forms.NumericUpDown.Minimum** property is set, the **System.Windows.Forms.NumericUpDown.Maximum** property is evaluated and the

System.Windows.Forms.NumericUpDown.UpdateEditText method is called. If the new **System.Windows.Forms.NumericUpDown.Minimum** property value is greater than the

System.Windows.Forms.NumericUpDown.Maximum property value, the **System.Windows.Forms.NumericUpDown.Maximum** value is set equal to the **System.Windows.Forms.NumericUpDown.Minimum** value. If the **System.Windows.Forms.NumericUpDown.Value** is less than the new **System.Windows.Forms.NumericUpDown.Minimum** value, the **System.Windows.Forms.NumericUpDown.Value** property is also set equal to the **System.Windows.Forms.NumericUpDown.Minimum** value.

1 *uuu) Name*
 2 *vvv) Parent*
 3 *www) ParentForm*
 4 *xxx) PreferredHeight*
 5 *yyy) ProductName*
 6 *zzz) ProductVersion*
 7 *aaaa) ReadOnly*
 8 *bbbb) RecreatingHandle*
 9 *cccc) Region*
 10 *dddd) RenderRightToLeft*
 11 *eeee) ResizeRedraw*
 12 *ffff) Right*
 13 *gggg) RightToLeft*
 14 *hhhh) ShowFocusCues*
 15 *iiii) ShowKeyboardCues*
 16 *jjjj) Site*
 17 *kkkk) Size*
 18 *llll) TabIndex*
 19 *mmmm)TabStop*
 20 *nnnn) Tag*
 21 *oooo) Text*
 22 *pppp) ToString*

1
2
3 *Description*

4 The text displayed in the control.

5 *qqqq) TextAlign*

6 *rrrr) ThousandsSeparator*

7 *ssss) ToString*
8
9

10 *Description*

11 Gets or sets a value indicating whether a thousands separator is displayed in the
12 up-down control when appropriate.

13 When the
14 **System.Windows.Forms.NumericUpDown.ThousandsSeparator** property
15 is set, the **System.Windows.Forms.NumericUpDown.UpdateEditText**
16 method is called to update the up-down control's display to the new format.

17 *tttt) Top*

18 *uuuu) TopLevelControl*

19 *vvvv) UpDownAlign*

20 *www)UserEdit*

21 *xxxx) Value*

22 *yyyy) ToString*
23

24 *Description*

25 Gets or sets the value assigned to the up-down control.

When the **System.Windows.Forms.NumericUpDown.Value** property is set, the new value is validated to be between the **System.Windows.Forms.NumericUpDown.Minimum** and **System.Windows.Forms.NumericUpDown.Maximum** values. Following this, the **System.Windows.Forms.NumericUpDown.UpdateEditText** method is called to update the up-down control's display with the new value in the appropriate format.

- zzzz) Visible*
- aaaaa) VScroll*
- bbbbbb) Width*
- cccccc) WindowTarget*
- dddddd) ToString*

Description

Occurs when the **System.Windows.Forms.NumericUpDown.Value** property has been changed in some way.

For the **System.Windows.Forms.NumericUpDown.OnValueChanged(System.EventArgs)** event to occur, the **System.Windows.Forms.NumericUpDown.Value** property may be changed in code, by the user typing in a new value, or by clicking the up or down button.

- eeeeee) BeginInit*

[C#]	public	void	BeginInit();
[C++]	public: __sealed	void	BeginInit();
[VB]	NotOverridable	Public Sub	BeginInit()
[JScript]	public	function	BeginInit();

Description

Handles tasks required when the control is being initialized.

fffff) CreateAccessibilityInstance

[C#] protected override AccessibleObject CreateAccessibilityInstance();

[C++] protected: AccessibleObject* CreateAccessibilityInstance();

[VB] Overrides Protected Function CreateAccessibilityInstance() As

AccessibleObject

[JScript] protected override function CreateAccessibilityInstance() :

AccessibleObject;

Description

ggggg) DownButton

[C#] public override void DownButton();

[C++] public: void DownButton();

[VB] Overrides Public Sub DownButton()

[JScript] public override function DownButton();

Description

Decrements the value of the up-down control.

When the **System.Windows.Forms.NumericUpDown.DownButton** method is called, either in code or by the click of the down button, the new value is validated and the control updated with the new value in the appropriate format.

Specifically, if **System.Windows.FormsUpDownBase.UserEdit** is set to **true**, **System.Windows.Forms.NumericUpDown.ParseEditText** is called prior to validating or updating the value. The value is then validated to be between the **System.Windows.Forms.NumericUpDown.Minimum** and **System.Windows.Forms.NumericUpDown.Maximum** values and the **System.Windows.Forms.NumericUpDown.UpdateEditText** method is called.

hhhhh)EndInit

[C#]	public	void	EndInit();
[C++]	public:	__sealed	void EndInit();
[VB]	NotOverridable	Public	Sub EndInit()
[JScript]	public	function	EndInit();

Description

Called when initialization of the control is complete.

iiii) OnTextBoxKeyPress

[C#]	protected	override	void	OnTextBoxKeyPress(object	source,
				KeyPressEventArgs	e);
[C++]	protected:	void	OnTextBoxKeyPress(Object*	source,	KeyPressEventArgs*
				e);	
[VB]	Overrides	Protected	Sub	OnTextBoxKeyPress(ByVal	source As Object,
	ByVal		e	As	KeyPressEventArgs)
[JScript]	protected	override	function	OnTextBoxKeyPress(source :	Object, e :
				KeyPressEventArgs);	

Description

Restricts the entry of characters to digits (including hex), the negative sign, the decimal point, and editing keystrokes (backspace).

jjjjj) OnValueChanged

[C#] protected virtual void OnValueChanged(EventArgs e);

[C++] protected: virtual void OnValueChanged(EventArgs* e);

[VB] Overridable Protected Sub OnValueChanged(ByVal e As EventArgs)

[JScript] protected function OnValueChanged(e : EventArgs);

Description

Raises the **System.Windows.Forms.NumericUpDown.ValueChanged** event.

Raising an event invokes the event handler through a delegate. For more information, see . An **System.EventArgs** that contains the event data.

kkkkk) ParseEditText

[C#] protected void ParseEditText();

[C++] protected: void ParseEditText();

[VB] Protected Sub ParseEditText()

[JScript] protected function ParseEditText();

Description

Converts the text displayed in the up-down control to a numeric value and evaluates it.

If **System.Windows.FormsUpDownBase.UserEdit** is set to **true** , the text displayed is converted to a numeric value so it may be validated to be between the **System.Windows.Forms.NumericUpDown.Minimum** and **System.Windows.Forms.NumericUpDown.Maximum** property values.

IIII) ToString

[C#]	public	override	string	ToString();
[C++]	public:		String*	ToString();
[VB]	Overrides	Public	Function ToString()	As String
[JScript]	public	override	function ToString()	: String;

Description

Provides some interesting info about this control in String form.

mmmmm) UpButton

[C#]	public	override	void	UpButton();
[C++]	public:		void	UpButton();
[VB]	Overrides	Public	Sub	UpButton()
[JScript]	public	override	function	UpButton();

Description

Increments the value of the up-down control.

When the **System.Windows.Forms.NumericUpDown.UpButton** method is called, either in code or by the click of the up button, the new value is validated and the control is updated with the new value in the appropriate format. Specifically, if **System.Windows.Forms.UpDownBase.UserEdit** is set to **true** , **System.Windows.Forms.NumericUpDown.ParseEditText** is called prior to validating or updating the value. The value is then validated to be between the **System.Windows.Forms.NumericUpDown.Minimum** and **System.Windows.Forms.NumericUpDown.Maximum** values and the

System.Windows.Forms.NumericUpDown.UpdateEditText method is called.

nnnnn)UpdateEditText

[C#]	protected	override	void	UpdateEditText();
[C++]	protected:		void	UpdateEditText();
[VB]	Overrides	Protected	Sub	UpdateEditText()
[JScript]	protected	override	function	UpdateEditText();

Description

Displays the current value of the up-down control in the appropriate format.

If the up-down control is not being initialized, the current value is validated to be between the **System.Windows.Forms.NumericUpDown.Minimum** and **System.Windows.Forms.NumericUpDown.Maximum** values and is converted to the correct format for display in the control.

ooooo) ValidateEditText

[C#]	protected	override	void	ValidateEditText();
[C++]	protected:		void	ValidateEditText();
[VB]	Overrides	Protected	Sub	ValidateEditText()
[JScript]	protected	override	function	ValidateEditText();

Description

Validates and updates the text displayed in the up-down control.

The **System.Windows.Forms.NumericUpDown.ValidateEditText** method calls the **System.Windows.Forms.NumericUpDown.ParseEditText** and **System.Windows.Forms.NumericUpDown.UpdateEditText** methods to validate and update the display of the up-down control.

1 **ListBox.ObjectCollection** class (**System.Windows.Forms**)

2 *a) WndProc*

5 *Description*

6 Represents the collection of items in a **System.Windows.Forms.ListBox** .

7 The **System.Windows.Forms.ListBox.ObjectCollection** class stores the
8 items displayed in the **System.Windows.Forms.ListBox** . There are two other
9 collections defined within the **System.Windows.Forms.ListBox** class that
10 enable you to determine what items are selected within this collection. The
11 **System.Windows.Forms.ListBox.SelectedObjectCollection** class provides
12 properties and methods for determining what items are selected within the
13 **System.Windows.Forms.ListBox.ObjectCollection** , while the
14 **System.Windows.Forms.ListBox.SelectedIndexCollection** class enables
15 you to determine what indices within the
16 **System.Windows.Forms.ListBox.ObjectCollection** are selected.

13 *b) ListBox.ObjectCollection*

14 *Example Syntax:*

15 *c) WndProc*

17 [C#] public **ListBox.ObjectCollection**(**ListBox** owner);

18 [C++] public: **ObjectCollection**(**ListBox*** owner);

19 [VB] Public Sub New(**ByVal** owner As **ListBox**)

20 [JScript] public function **ListBox.ObjectCollection**(owner : **ListBox**); Initializes a
21 new instance of **System.Windows.Forms.ListBox.ObjectCollection** .

23 *Description*

24 Initializes a new instance of
25 **System.Windows.Forms.ListBox.ObjectCollection** .

1 You cannot create an instance of this class without associating it with a
2 **System.Windows.Forms.ListBox** control. The
3 **System.Windows.Forms.ListBox** that owns the collection.

4 *d) **ListBox.ObjectCollection***

5 *Example Syntax:*

6 *e) **WndProc***

7 [C#] public ListBox.ObjectCollection(ListBox owner, object[] value);

8 [C++] public: ObjectCollection(ListBox* owner, Object* value __gc[]);

9 [VB] Public Sub New(ByVal owner As ListBox, ByVal value() As Object)

10 [JScript] public function ListBox.ObjectCollection(owner : ListBox, value :
11 Object[]);

12
13 *Description*

14 Initializes a new instance of
15 **System.Windows.Forms.ListBox.ObjectCollection** containing an array of
16 objects.

17 You cannot create an instance of this class without associating it with a
18 **System.Windows.Forms.ListBox** control. The
19 **System.Windows.Forms.ListBox** that owns the collection. An array of objects
20 to add to the collection.

21 *f) **ListBox.ObjectCollection***

22 *Example Syntax:*

23 *g) **WndProc***

24 [C#] public ListBox.ObjectCollection(ListBox owner, ListBox.ObjectCollection
25 value);

[C++] public: ObjectCollection(ListBox* owner, ListBox.ObjectCollection*

```

1 value);
2 [VB] Public Sub New(ByVal owner As ListBox, ByVal value As
3 ListBox.ObjectCollection)
4 [JScript] public function ListBox.ObjectCollection(owner : ListBox, value :
5 ListBox.ObjectCollection);
6

```

Description

Initializes a new instance of **System.Windows.Forms.ListBox.ObjectCollection** based on another **System.Windows.Forms.ListBox.ObjectCollection** .

You cannot create an instance of this class without associating it with a **System.Windows.Forms.ListBox** control. This version of the constructor enables you to use the items specified in an existing instance of the **System.Windows.Forms.ListBox.ObjectCollection** class to add items to the collection when it is created. You can use this constructor to use the items specified in another **System.Windows.Forms.ListBox** control with this collection. The **System.Windows.Forms.ListBox** that owns the collection. A **System.Windows.Forms.ListBox.ObjectCollection** from which the contents are copied to this collection.

- h) Count*
- i) WndProc*

[C#]	public	int	Count	{get;}
[C++]	public:	__property	int	get_Count();
[VB]	Public	ReadOnly	Property	Count As Integer
[JScript]	public	function	get	Count() : int;

Description

Gets the number of items in the collection.

This property enables you to determine the number of items in the **System.Windows.Forms.ListBox** . You can then use this value when you are looping through the values of the collection and you need to provide a number of iterations to perform the loop.

j) *IsReadOnly*

k) *WndProc*

[C#] public bool IsReadOnly {get;}

[C++] public: __property bool get_IsReadOnly();

[VB] Public ReadOnly Property IsReadOnly As Boolean

[JScript] public function get IsReadOnly() : Boolean;

Description

Gets a value indicating whether the collection is read-only.

This property is always **false** for this collection.

l) *Item*

m) *WndProc*

[C#] public virtual object this[int index] {get; set;}

[C++] public: __property virtual Object* get_Item(int index);public: __property

virtual void set_Item(int index, Object*);

[VB] Overridable Public Default Property Item(ByVal index As Integer) As

Object

[JScript] returnValue =

ObjectCollectionObject.Item(index);ObjectCollectionObject.Item(index) =

returnValue;

Description

Gets or sets the item at the specified index within the collection.

You can use this method to obtain the **System.Windows.Forms.ListBox** item stored at a specific location within the collection. To determine the index of a specific item within the collection, use the **System.Windows.Forms.ListBox.ObjectCollection.IndexOf(System.Object)** method. The index of the item in the collection to get or set.

n) Add

[C#]	public	int	Add(object	item);
[C++]	public:	int	Add(Object*	item);
[VB]	Public	Function	Add(ByVal item	As Object) As Integer
[JScript]	public	function	Add(item	: Object) : int;

Description

Adds an item to the list of items for a **System.Windows.Forms.ListBox**.
Return Value: The zero-based index of the item in the collection.

If the **System.Windows.Forms.ListBox.Sorted** property of the **System.Windows.Forms.ListBox** is set to **true**, the item is inserted into the list alphabetically. Otherwise, the item is inserted at the end of the list. To insert an item into the list box at a specific position, use the **System.Windows.Forms.ListBox.ObjectCollection.Insert(System.Int32, System.Object)** method. To add a set of items to the list box in a single operation, use the **System.Windows.Forms.ListBox.ObjectCollection.AddRange(System.Windows.Forms.ListBox.ObjectCollection)** method. If you want to use the **System.Windows.Forms.ListBox.ObjectCollection.Add(System.Object)** method to add a large number of items to the list, use the **System.Windows.Forms.ListBox.BeginUpdate** and **System.Windows.Forms.ListBox.EndUpdate** methods to prevent the **System.Windows.Forms.ListBox** from repainting each time an item is added to the list until all items are added to the list. When adding items to a **System.Windows.Forms.ListBox**, it is more efficient to sort the items first

and then add new items. An object representing the item to add to the collection.

o) AddRange

```
[C#]      public      void      AddRange(object[]      items);
[C++]      public:      void      AddRange(Object*      items      __gc[]);
[VB]      Public      Sub      AddRange(ByVal      items()      As      Object)
[JScript] public function AddRange(items : Object[]); Adds a group of items to the
list      of      items      for      a      System.Windows.Forms.ListBox .
```

Description

Adds an array of items to the list of items for a **System.Windows.Forms.ListBox** .

This method removes all existing items from the list box before inserting the new items. If the **System.Windows.Forms.ListBox.Sorted** property of the **System.Windows.Forms.ListBox** is set to **true** , the items are inserted into the list alphabetically. Otherwise, the items are inserted in the order that they occur within the array. This method is typically passed an array of **System.String** objects, but an array of any type of object can be passed to this method. When an object is added to the collection, the **System.Windows.Forms.ListBox** first checks to see if the **System.Windows.Forms.ListControl.DisplayMember** property of the **System.Windows.Forms.ListControl** class has the name of a member from the object specified to reference when obtaining the item text. If the **System.Windows.Forms.ListControl.DisplayMember** property does not have a member specified, the **System.Windows.Forms.ListBox** then calls the **System.Object.ToString** method of the object to obtain the text to display in the list. When using this method to add items to the **System.Windows.Forms.ListBox** , you do not need to call the **System.Windows.Forms.ListBox.BeginUpdate** and **System.Windows.Forms.ListBox.EndUpdate** methods to optimize performance. When adding items to a **System.Windows.Forms.ListBox** , it is more efficient to sort the items first and then add new items. You can use this method to add a group of items to the list or to reuse the items stored in a different **System.Windows.Forms.ListBox** . An array of objects to add to the list.

p) *AddRange*

```
[C#]      public      void      AddRange(ListBox.ObjectCollection      value);
[C++]     public:     void      AddRange(ListBox.ObjectCollection*      value);
[VB]      Public Sub  AddRange(ByVal value As ListBox.ObjectCollection)
[JavaScript] public function AddRange(value : ListBox.ObjectCollection);
```

Description

Adds the items of an existing **System.Windows.Forms.ListBox.ObjectCollection** to the list of items in a **System.Windows.Forms.ListBox**.

This method removes all existing items from the list box before inserting the new items. If the **System.Windows.Forms.ListBox.Sorted** property of the **System.Windows.Forms.ListBox** is set to **true**, the items are inserted into the list alphabetically. Otherwise, the items are inserted in the order that they occur within the array. This method is typically passed an array of **System.String** objects, but an array of any type of object can be passed to this method. When an object is added to the collection, the **System.Windows.Forms.ListBox** first checks to see if the **System.Windows.Forms.ListControl.DisplayMember** property of the **System.Windows.Forms.ListControl** class has the name of a member from the object specified to reference when obtaining the item text. If the **System.Windows.Forms.ListControl.DisplayMember** property does not have a member specified, the **System.Windows.Forms.ListBox** then calls the **System.Object.ToString** method of the object to obtain the text to display in the list. A **System.Windows.Forms.ListBox.ObjectCollection** to load into this collection.

q) *Clear*

```
[C#]      public      virtual      void      Clear();
[C++]     public:     virtual      void      Clear();
[VB]      Overridable      Public      Sub      Clear()
[JavaScript]      public      function      Clear();
```

Description

Removes all items from the collection.

When you remove items from the list, all information about the deleted items is lost. To remove a single item from the **System.Windows.Forms.ListBox**, use the

System.Windows.Forms.ListBox.ObjectCollection.Remove(System.Object) or

System.Windows.Forms.ListBox.ObjectCollection.RemoveAt(System.Int32) method.

r) Contains

[C#] public bool Contains(object value);

[C++] public: __sealed bool Contains(Object* value);

[VB] NotOverridable Public Function Contains(ByVal value As Object) As Boolean

[JScript] public function Contains(value : Object) : Boolean;

Description

Determines whether the specified item is located within the collection.

Return Value: **true** if the item is located within the collection; otherwise, **false**.

The

System.Windows.Forms.ListBox.ObjectCollection.Contains(System.Object) method enables you to determine whether an object is a member of the collection. Once you know that the item is located within the collection, you can use the

System.Windows.Forms.ListBox.ObjectCollection.IndexOf(System.Object) method to determine where the item is located within the collection. An object representing the item to locate in the collection.

s) *CopyTo*

[C#] public void CopyTo(object[] dest, int arrayIndex);
[C++] public: void CopyTo(Object* dest __gc[], int arrayIndex);
[VB] Public Sub CopyTo(ByVal dest() As Object, ByVal arrayIndex As Integer)
[JScript] public function CopyTo(dest : Object[], arrayIndex : int); Copies the entire collection into an existing array at a specified location within the array.

Description

Copies the entire collection into an existing array of objects at a specified location within the array.

You can use this method to combine the items from multiple collections into a single array. You can then use this array to populate the contents of another **System.Windows.Forms.ListBox** control using the **System.Windows.Forms.ListBox.ObjectCollection.AddRange(System.Windows.Forms.ListBox.ObjectCollection)** method of the **System.Windows.Forms.ListBox.ObjectCollection** class. The object array in which the items from the collection are copied to. The location within the destination array to copy the items from the collection to.

t) *GetEnumerator*

[C#] public IEnumerator GetEnumerator();
[C++] public: __sealed IEnumerator* GetEnumerator();
[VB] NotOverridable Public Function GetEnumerator() As IEnumerator
[JScript] public function GetEnumerator() : IEnumerator;

Description

Returns an enumerator to use to iterate through the item collection.

Return Value: An **System.Collections.IEnumerator** object that represents the item collection.

u) *IndexOf*

```
[C#]      public      int      IndexOf(object      value);
[C++]     public:     __sealed   int      IndexOf(Object*      value);
[VB] NotOverridable Public Function IndexOf(ByVal value As Object) As Integer
[JScript] public      function   IndexOf(value      :      Object)      :      int;
```

Description

Returns the index within the collection of the specified item.

Return Value: The zero-based index where the item is located within the collection; otherwise, negative one (-1).

The

System.Windows.Forms.ListBox.ObjectCollection.IndexOf(System.Object) method enables you to determine where an item is located within the collection. To determine whether an item is located within the collection before calling this method, use the **System.Windows.Forms.ListBox.ObjectCollection.Contains(System.Object)** method. An object representing the item to locate in the collection.

v) *Insert*

```
[C#]      public      void      Insert(int      index,      object      item);
[C++]     public:     __sealed   void      Insert(int      index,      Object*      item);
[VB] NotOverridable Public Sub Insert(ByVal index As Integer, ByVal item As Object)
[JScript] public      function   Insert(index      :      int,      item      :      Object);
```

Description

Inserts an item into the list box at the specified index.

This method enables you to insert an item at a specific position within the **System.Windows.Forms.ListBox**. If the **System.Windows.Forms.ListBox.Sorted** property of the **System.Windows.Forms.ListBox** is set to true, the item is added in the correct position within the sorted list regardless of the values specified in the *index* parameter. When an object is added to the collection, the **System.Windows.Forms.ListBox** first checks to see if the **System.Windows.Forms.ListControl.DisplayMember** property of the **System.Windows.Forms.ListControl** class has the name of a member from the object specified to reference when obtaining the item text. If the **System.Windows.Forms.ListControl.DisplayMember** property does not have a member specified, the **System.Windows.Forms.ListBox** then calls the **System.Object.ToString** method of the object to obtain the text to display in the list. The zero-based index location where the item is inserted. An object representing the item to insert.

w) *Remove*

```
[C#]          public          void          Remove(object          value);
[C++]         public:         __sealed      void          Remove(Object*          value);
[VB]  NotOverridable  Public  Sub  Remove(ByVal  value  As  Object)
[JScript]      public          function      Remove(value          :          Object);
```

Description

Removes the specified object from the collection.

When you remove an item from the list, the indices change for subsequent items in the list. All information about the removed item is deleted. You can use this method to remove a specific item from the list by specifying the actual item to remove from the list. To specify the index of the item to remove instead of the item itself, use the

System.Windows.Forms.ListBox.ObjectCollection.RemoveAt(System.Int32) method. To remove all items from the list, use the **System.Windows.Forms.ListBox.ObjectCollection.Clear** method. An object representing the item to remove from the collection.

x) *RemoveAt*

```
[C#]          public          void          RemoveAt(int          index);
[C++]         public:         __sealed      void          RemoveAt(int          index);
[VB] NotOverridable Public Sub RemoveAt(ByVal index As Integer)
[JavaScript]   public         function      RemoveAt(index      :      int);
```

Description

Removes the item at the specified index within the collection.

When you remove an item from the list, the indices change for subsequent items in the list. All information about the removed item is deleted. You can use this method to remove a specific item from the list by specifying the index of the item to remove from the list. To specify the item to remove instead of the index to the item, use the

System.Windows.Forms.ListBox.ObjectCollection.Remove(System.Object) method. To remove all items from the list, use the **System.Windows.Forms.ListBox.ObjectCollection.Clear** method. The zero-based index of the item to remove.

y) *ICollection.CopyTo*

```
[C#]          void          ICollection.CopyTo(Array          dest,          int          index);
[C++]         void          ICollection::CopyTo(Array*          dest,          int          index);
[VB] Sub CopyTo(ByVal dest As Array, ByVal index As Integer) Implements
ICollection.CopyTo
[JavaScript] function ICollection.CopyTo(dest : Array, index : int);
```

z) *IList.Add*

```
[C#]          int          IList.Add(object          item);
```

1 [C++] int IList::Add(Object* item);

2 [VB] Function Add(ByVal item As Object) As Integer Implements IList.Add

3 [JScript] function IList.Add(item : Object) : int;

4 CheckedListBox.ObjectCollection class (System.Windows.Forms)

5 a) *ToString*

8 *Description*

9 Represents the collection of items in a
10 **System.Windows.Forms.CheckedListBox** .

11 The collection is accessed from the parent control,
12 **System.Windows.Forms.CheckedListBox** , by the
13 **System.Windows.Forms.CheckedListBox.Items** property. To create a
14 collection of objects to display in the
15 **System.Windows.Forms.CheckedListBox** control, you can add or remove
16 the items individually by using the
17 **System.Windows.Forms.CheckedListBox.ObjectCollection.Add(System.
18 Object,System.Boolean)** and
19 **System.Windows.Forms.ListBox.ObjectCollection.Remove(System.Object)**
20 methods.

17 b) *CheckedListBox.ObjectCollection*

18 *Example Syntax:*

19 c) *ToString*

21 [C#] public CheckedListBox.ObjectCollection(CheckedListBox owner);

22 [C++] public: ObjectCollection(CheckedListBox* owner);

23 [VB] Public Sub New(ByVal owner As CheckedListBox)

24 [JScript] public function CheckedListBox.ObjectCollection(owner :

25 CheckedListBox);

Initializes a new instance of the **System.Windows.Forms.CheckedListBox.ObjectCollection** class. The **System.Windows.Forms.CheckedListBox** that owns the collection.

g) Add

[JScript] public function Add(item : Object, isChecked : Boolean) : int; Adds an item to the list of items for a **System.Windows.Forms.CheckedListBox**.

This method adds an item to the list. For a list, the item is added to the end of the existing list of items. For a sorted checked list box, the item is inserted into the list according to its sorted position. A **SystemException** occurs if there is insufficient space available to store the new item. An object representing the item to add to the collection. **true** to check the item; otherwise, **false**.

h) *Add*

[C#] public int Add(object item, CheckState check);

[C++] public: int Add(Object* item, CheckState check);

[VB] Public Function Add(ByVal item As Object, ByVal check As CheckState)

As Integer

[JScript] public function Add(item : Object, check : CheckState) : int;

Description

Adds an item to the list of items for a

System.Windows.Forms.CheckedListBox, specifying the object to add and the initial checked value.

Return Value: The index of the newly added item.

This method adds an item to the checked list box. For an unsorted checked list box, the item is added to the end of the existing list of items. For a sorted checked list box, the item is inserted into the list according to its sorted position.

A **SystemException** occurs if there is insufficient space available to store the new item. An object representing the item to add to the collection. The initial **System.Windows.Forms.CheckState** for the checked portion of the item.

ComboBox.ObjectCollection class (System.Windows.Forms)

a) *ToString*

Description

Represents the collection of items in a **System.Windows.Forms.ComboBox**.

The **System.Windows.Forms.ComboBox.ObjectCollection** class encapsulates the items in the **System.Windows.Forms.ComboBox**. The object collection of a combo box can be used to manage many types of objects, including strings, images, and custom business objects.

b) *ComboBox.ObjectCollection*

Example Syntax:

c) *ToString*

```
[C#]      public      ComboBox.ObjectCollection(ComboBox      owner);
[C++]      public:      ObjectCollection(ComboBox*      owner);
[VB]      Public      Sub      New(ByVal      owner      As      ComboBox)
[JScript] public function ComboBox.ObjectCollection(owner : ComboBox);
```

Description

Initializes a new instance of

System.Windows.Forms.ComboBox.ObjectCollection .

An instance of this class cannot be created without associating it with a **System.Windows.Forms.ComboBox** control. The **System.Windows.Forms.ComboBox** that owns this object collection.

d) *Count*

e) *ToString*

```
[C#]      public      int      Count      {get;}
[C++]      public:      __property      int      get_Count();
[VB]      Public      ReadOnly      Property      Count      As      Integer
[JScript] public function get      Count()      :      int;
```

Description

Gets the number of items in the collection.

This property enables you to determine the number of items in the **System.Windows.Forms.ComboBox** . You can use this value when looping through the values of the collection.

f) IsReadOnly

g) ToString

```
[C#]      public      bool      IsReadOnly      {get;}
```

```
[C++] public: __property bool get IsReadOnly();
```

[VB]	Public	ReadOnly	Property	IsReadOnly	As	Boolean
------	--------	----------	----------	------------	----	---------

```
[JScript] public function get IsReadOnly() : Boolean;
```

Description

Gets a value indicating whether this collection can be modified.

h) Item

i) ToString

```
[C#] public virtual object this[int index] {get; set;}
```

```
[C++] public: __property virtual Object* get Item(int index);public: property
```

```
virtual void set Item(int index, Object*);
```

```
[VB] Overridable Public Default Property Item(ByVal index As Integer) As
Object
```

```
[JScript] return Value =
```

```
ObjectCollectionObject.Item(index);ObjectCollectionObject.Item(index) =
```

```
return Value;
```


Description

Retrieves the item at the specified index within the collection.

You can use this method to obtain the item at the specified location within the collection. You can use

System.Windows.Forms.ComboBox.ObjectCollection.IndexOf(System.Object) to find the location of an item, or you can use the index return from the **System.Windows.Forms.ComboBox.ObjectCollection.Add(System.Object)** method. The index of the item in the collection to retrieve.

j) Add

[C#]	public	int	Add(object	item);
[C++]	public:	int	Add(Object*	item);
[VB]	Public	Function	Add(ByVal item	As Object) As Integer
[JScript]	public	function	Add(item	: Object) : int;

Description

Adds an item to the list of items for a **System.Windows.Forms.ComboBox**.

Return Value: The zero-based index of the item in the collection.

This method adds an item to the combo box. If the **System.Windows.Forms.ComboBox.Sorted** property of the **System.Windows.Forms.ComboBox** is set to **true**, the item is inserted into the list alphabetically. Otherwise, the item is inserted at the end of the list. An object representing the item to add to the collection.

k) AddRange

[C#]	public	void	AddRange(object[]	items);
[C++]	public:	void	AddRange(Object*	items __gc[]);
[VB]	Public	Sub	AddRange(ByVal items()	As Object)

[JScript] public function AddRange(items : Object[]);

Description

Adds an array of items to the list of items for a **System.Windows.Forms.ComboBox** .

If the **System.Windows.Forms.ComboBox.Sorted** property of the **System.Windows.Forms.ComboBox** is set to true, the items are inserted into the list alphabetically. Otherwise, the items are inserted in the order they occur within the array. This method is typically passed an array of **System.String** objects, but an array of any type of object can be passed to this method. When an object is added to the collection, the method calls the object's **System.Object.ToString** method to obtain the string to display in the list. When using this method to add items to the collection, you do not need to call the **System.Windows.Forms.ComboBox.BeginUpdate** and **System.Windows.Forms.ComboBox.EndUpdate** methods to optimize performance. An array of objects to add to the list.

1) Clear

[C#] public void Clear();

[C++] public: __sealed void Clear();

[VB] NotOverridable Public Sub Clear()

[JScript] public function Clear();

Description

Removes all items from the **System.Windows.Forms.ComboBox** .

When you remove items from the list, all information about the deleted items is lost. To remove a single item from the **System.Windows.Forms.ComboBox** , use the

System.Windows.Forms.ComboBox.ObjectCollection.Remove(System.Object) or

System.Windows.Forms.ComboBox.ObjectCollection.RemoveAt(System.Int32) method.

m) Contains

```
[C#]      public      bool      Contains(object      value);
[C++]     public:     __sealed    bool      Contains(Object*      value);
[VB] NotOverridable Public Function Contains(ByVal value As Object) As
Boolean
[JScript] public function Contains(value : Object) : Boolean;
```

Description

Determines if the specified item is located within the collection.

Return Value: **true** if the item is located within the collection; otherwise, **false**.

The

System.Windows.Forms.ComboBox.ObjectCollection.Contains(System.Object) method enables you to determine if an object is a member of the collection. Once you know that the item is located within the collection, you can use the

System.Windows.Forms.ComboBox.ObjectCollection.IndexOf(System.Object) method to determine where the item is located within the collection. An object representing the item to locate in the collection.

n) CopyTo

```
[C#]      public      void      CopyTo(object[]      dest,      int      arrayIndex);
[C++]     public:     void      CopyTo(Object*      dest      __gc[],      int      arrayIndex);
[VB] Public Sub CopyTo(ByVal dest() As Object, ByVal arrayIndex As Integer)
[JScript] public function CopyTo(dest : Object[], arrayIndex : int);
```

Description

Copies the entire collection into an existing array of objects at a specified location within the array.

You can use this method to combine the items from multiple collections into a single array. You can then use this array to populate the contents of another **System.Windows.Forms.ComboBox** control using the **System.Windows.Forms.ComboBox.ObjectCollection.AddRange(System.Object[])** method of the **System.Windows.Forms.ComboBox.ObjectCollection** class. The object array to copy the collection to. The location in the destination array to copy the collection to.

o) GetEnumerator

```
[C#]          public          IEnumerator          GetEnumerator();
[C++]        public:          __sealed          IEnumerator*          GetEnumerator();
[VB] NotOverridable Public Function GetEnumerator() As IEnumerator
[JScript]    public          function          GetEnumerator()          :          IEnumerator;
```

Description

Returns an enumerator that can be used to iterate through the item collection.
Return Value: An **System.Collections.IEnumerator** object that represents the item collection.

p) IndexOf

```
[C#]          public          int          IndexOf(object          value);
[C++]        public:          __sealed          int          IndexOf(Object*          value);
[VB] NotOverridable Public Function IndexOf(ByVal value As Object) As Integer
[JScript]    public          function          IndexOf(value          :          Object)          :          int;
```

Description

Retrieves the index within the collection of the specified item.
Return Value: The zero-based index where the item is located within the collection; otherwise, -1.

The

System.Windows.Forms.ComboBox.ObjectCollection.IndexOf(System.Object) method enables you to determine where an item is located within the collection. To determine if an item is located within the collection before calling this method, use the

System.Windows.Forms.ComboBox.ObjectCollection.Contains(System.Object) method. An object representing the item to locate in the collection.

q) Insert

[C#] public void Insert(int index, object item);

[C++] public: __sealed void Insert(int index, Object* item);

[VB] NotOverridable Public Sub Insert(ByVal index As Integer, ByVal item As Object)

[JScript] public function Insert(index : int, item : Object);

Description

Inserts an item into the collection at the specified index.

Return Value: The zero-based index of the newly added item.

If the **System.Windows.Forms.ComboBox.sorted** property of the **System.Windows.Forms.ComboBox** is true, the *index* parameter is ignored. When an object is added to the collection, the

System.Windows.Forms.ComboBox calls the object's

System.Object.ToString method to obtain the string to display in the list.

The zero-based index location where the item is inserted. An object representing the item to insert.

r) Remove

[C#] public void Remove(object value);

[C++] public: __sealed void Remove(Object* value);

[VB] NotOverridable Public Sub Remove(ByVal value As Object)

[JScript] public function Remove(value : Object);

Description

Removes the specified item from the **System.Windows.Forms.ComboBox**.
The **System.Object** to remove from the list.

s) *RemoveAt*

[C#] public void RemoveAt(int index);

[C++] public: __sealed void RemoveAt(int index);

[VB] NotOverridable Public Sub RemoveAt(ByVal index As Integer)

[JScript] public function RemoveAt(index : int);

Description

Removes an item from the **System.Windows.Forms.ComboBox** at the
specified index. The index of the item to remove.

t) *ICollection.CopyTo*

[C#] void ICollection.CopyTo(Array dest, int index);

[C++] void ICollection::CopyTo(Array* dest, int index);

[VB] Sub CopyTo(ByVal dest As Array, ByVal index As Integer) Implements
ICollection.CopyTo

[JScript] function ICollection.CopyTo(dest : Array, index : int);

u) *IList.Add*

[C#] int IList.Add(object item);

[C++] int IList::Add(Object* item);


```

1 ITypeDescriptorContext, ByVal sourceType As Type) As Boolean
2 [JScript] public override function CanConvertFrom(context :
3 ITypeDescriptorContext, sourceType : Type) : Boolean; Returns whether this
4 converter can convert an object in the given source type to the native type of the
5 converter.

```

7 *Description*

8 Returns a value indicating whether this converter can convert an object of the
9 specified source type to the native type of the converter using the specified
context.

Return Value: **true** if this converter can perform the conversion; otherwise,
false .

11 The *sourceType* is expected to be a **System.String** . A
System.ComponentModel.ITypeDescriptorContext that provides
12 information about the context of a type converter. The **System.Type** that
13 represents the type you wish to convert from.

14 *e) ConvertFrom*

```

15
16 [C#] public override object ConvertFrom(ITypeDescriptorContext context,
17 CultureInfo culture, object value);
18 [C++] public: Object* ConvertFrom(ITypeDescriptorContext* context,
19 CultureInfo* culture, Object* value);
20 [VB] Overrides Public Function ConvertFrom(ByVal context As
21 ITypeDescriptorContext, ByVal culture As CultureInfo, ByVal value As Object)
22 As Object
23 [JScript] public override function ConvertFrom(context : ITypeDescriptorContext,
24 culture : CultureInfo, value : Object) : Object; Converts the given value to the
25 native type of the converter.

```


Description

Converts the specified object to the coverter's native type.

Return Value: An **System.Object** that represents the converted *value*.

Override this method to provide your own conversion requirements. A **System.ComponentModel.ITypeDescriptorContext** that provides information about the context of a type converter. The locale information for the conversion. The object to convert.

f) ConvertTo

```
[C#] public override object ConvertTo(ITypeDescriptorContext context,
CultureInfo culture, object value, Type destinationType);
```

```
[C++] public: Object* ConvertTo(ITypeDescriptorContext* context, CultureInfo*
culture, Object* value, Type* destinationType);
```

```
[VB] Overrides Public Function ConvertTo(ByVal context As
ITypeDescriptorContext, ByVal culture As CultureInfo, ByVal value As Object,
ByVal destinationType As Type) As Object
```

```
[JScript] public override function ConvertTo(context : ITypeDescriptorContext,
culture : CultureInfo, value : Object, destinationType : Type) : Object; Converts
the given value object to the specified destination type.
```

Description

Converts from the coverter's native type to a value of the destination type.

Return Value: An **System.Object** that represents the converted *value*.

Override this method to provide your own conversion requirements. A **System.ComponentModel.ITypeDescriptorContext** that provides information about the context of a type converter. The locale information for the conversion. The value to convert. The type to convert the object to.

OpenFileDialog class (System.Windows.Forms)

a) ToString

Description

Represents a common dialog box that displays the control that allows the user to open a file. This class cannot be inherited.

This class allows you to check if a file exists and open it. The **System.Windows.Forms.OpenFileDialog.ShowReadOnly** property determines if a read-only check box appears in the dialog box. The **System.Windows.Forms.OpenFileDialog.ReadOnlyChecked** property indicates whether the read-only check box is checked.

b) OpenFileDialog

Example Syntax:

c) ToString

[C#]	public	OpenFileDialog();
[C++]	public:	OpenFileDialog();
[VB]	Public	Sub New()
[JScript]	public function	OpenFileDialog();

d) AddExtension

e) CheckFileExists

f) ToString

Description

Gets or sets a value indicating whether the dialog box displays a warning if the user specifies a file name that does not exist.

g) CheckPathExists

h) Container

i) DefaultExt

j) DereferenceLinks

k) DesignMode

l) Events

m) FileName

n) FileNames

o) Filter

p) FilterIndex

q) InitialDirectory

r) Instance

s) Multiselect

t) ToString

Description

Gets or sets a value indicating whether the dialog box allows multiple files to be selected.

Use **System.Windows.Forms.FileDialog.FileNames** to access the fill list of selected file names.

- u) *Options*
- v) *ReadOnlyChecked*
- w) *ToString*

Description

Gets or sets a value indicating whether the read-only check box is selected.

The **System.Windows.Forms.OpenFileDialog.ReadOnlyChecked** state does not affect the read/write mode that **System.Windows.Forms.OpenFileDialog.OpenFile** uses to open a file selected in the dialog.

- x) *RestoreDirectory*
- y) *ShowHelp*
- z) *ShowReadOnly*
- aa) *ToString*

Description

Gets or sets a value indicating whether the dialog contains a read-only check box.

- bb) *Site*
- cc) *Title*
- dd) *ValidateNames*
- ee) *OpenFile*

```
[C#]          public          Stream          OpenFile();
```

[C++]	public:	Stream*	OpenFile();
[VB]	Public	Function	OpenFile() As Stream
[JScript]	public	function	OpenFile() : Stream;

Description

Opens the file selected by the user, with read-only permission. The file is specified by the **System.Windows.Forms.FileDialog.FileName** property.
Return Value: A **System.IO.Stream** that specifies the read-only file selected by the user.

The **System.Windows.Forms.OpenFileDialog.OpenFile** method is used to provide a facility to quickly open a file from the dialog box. The file is opened in read-only mode for security purposes. To open a file in a read/write mode, you must use another call such as **System.IO.FileStream**.

ff) Reset

[C#]	public	override	void	Reset();
[C++]	public:		void	Reset();
[VB]	Overrides	Public	Sub	Reset()
[JScript]	public	override	function	Reset();

Description

Resets all properties to their default values.

Orientation enumeration (System.Windows.Forms)

a) ToString

Description

Specifies the orientation of controls or elements of controls.

This enumeration is used by members such as
System.Windows.Forms.TrackBar.Orientation .

b) ToString

[C#]	public	const	Orientation	Horizontal;
[C++]	public:	const	Orientation	Horizontal;
[VB]	Public	Const	Horizontal	As Orientation
[JScript]	public	var	Horizontal	: Orientation;

Description

The control or element is oriented horizontally.

c) ToString

[C#]	public	const	Orientation	Vertical;
[C++]	public:	const	Orientation	Vertical;
[VB]	Public	Const	Vertical	As Orientation
[JScript]	public	var	Vertical	: Orientation;

Description

The control or element is oriented vertically.

OSFeature class (System.Windows.Forms)

a) ToString

Description

Provides operating-system specific feature queries.

Use the **static** instance of this class provided in the **System.Windows.Forms.OSFeature.Feature** property to query for operating system features. You cannot create an instance of this class.

b) ToString

[C#] public static readonly object LayeredWindows;

[C++] public: static Object* LayeredWindows;

[VB] Public Shared ReadOnly LayeredWindows As Object

[JScript] public static var LayeredWindows : Object;

Description

Represents the layered, top-level windows feature. This **static** field is read-only.

Layered windows are available only in Windows 2000. A layered window can be made transparent or translucent by the operating system.

c) ToString

[C#] public static readonly object Themes;

[C++] public: static Object* Themes;

[VB] Public Shared ReadOnly Themes As Object

[JScript] public static var Themes : Object;

Description

Represents the operating system themes feature. This **static** field is read-only.

d) OSFeature

Example Syntax:

e) *ToString*

[C#]	protected	OSFeature();
[C++]	protected:	OSFeature();
[VB]	Protected Sub	New()
[JScript]	protected function	OSFeature();

Description

Initializes a new instance of the **System.Windows.Forms.OSFeature** class.

This class cannot be instantiated. To query for operating system features, use the **static** instance of **System.Windows.Forms.OSFeature** provided in this class.

f) *Feature*

g) *ToString*

[C#]	public	static	OSFeature	Feature	{get;}
[C++]	public:	__property	static	OSFeature*	get_Feature();
[VB]	Public	Shared	ReadOnly	Property	Feature As OSFeature
[JScript]	public	static	function	get	Feature() : OSFeature;

Description

Represents the **static** instance of **System.Windows.Forms.OSFeature** to use for feature queries. This property is read-only.

Use this **static** property to query for operating system features. You cannot create an instance of this class.

h) *GetVersionPresent*

```
[C#] public override Version GetVersionPresent(object feature);  
[C++] public: Version* GetVersionPresent(Object* feature);  
[VB] Overrides Public Function GetVersionPresent(ByVal feature As Object) As  
Version  
[JScript] public override function GetVersionPresent(feature : Object) : Version;
```

Description

Retrieves the version of the specified feature currently available on the system.
Return Value: A **System.Version** representing the version of the specified operating system feature currently available on the system; or **null** if the feature cannot be found.

Use **System.Windows.Forms.OSFeature.feature**, the **static** instance of **System.Windows.Forms.OSFeature** provided in this class, to query for the version number of a feature. The feature whose version is requested.

OwnerDrawPropertyBag class (System.Windows.Forms)

a) *ToString*

Description

Class used to pass new font/color information around for "partial" ownerdraw list/treeview items.

b) *OwnerDrawPropertyBag*

Example Syntax:

c) *ToString*

```
[C#] public OwnerDrawPropertyBag();
```

1 [C++] public: OwnerDrawPropertyBag();

2 [VB] Public Sub New()

3 [JScript] public function OwnerDrawPropertyBag();

4 *d) BackColor*

5 *e) ToString*

7 [C#] public Color BackColor {get; set;}

8 [C++] public: __property Color get_BackColor();public: __property void
9 set_BackColor(Color);

10 [VB] Public Property BackColor As Color

11 [JScript] public function get BackColor() : Color;public function set
12 BackColor(Color);

14 *Description*

16 *f) Font*

17 *g) ToString*

19 [C#] public Font Font {get; set;}

20 [C++] public: __property Font* get_Font();public: __property void
21 set_Font(Font*);

22 [VB] Public Property Font As Font

23 [JScript] public function get Font() : Font;public function set Font(Font);

Description

h) ForeColor

i) ToString

[C#] public Color ForeColor {get; set;}

[C++] public: __property Color get_ForeColor();public: __property void
set_ForeColor(Color);

[VB] Public Property ForeColor As Color

[JScript] public function get ForeColor() : Color;public function set
ForeColor(Color);

Description

j) Copy

[C#] public static OwnerDrawPropertyBag Copy(OwnerDrawPropertyBag value);

[C++] public: static OwnerDrawPropertyBag* Copy(OwnerDrawPropertyBag*
value);

[VB] Public Shared Function Copy(ByVal value As OwnerDrawPropertyBag) As
OwnerDrawPropertyBag

[JScript] public static function Copy(value : OwnerDrawPropertyBag) :
OwnerDrawPropertyBag;

Description

Copies the bag. Always returns a valid ODPB object

Return Value: a shallow copy of the property bag Copies the bag. Always returns a valid ODPB object property bag to copy

k) IsEmpty

[C#] public virtual bool IsEmpty();

[C++] public: virtual bool IsEmpty();

[VB] Overridable Public Function IsEmpty() As Boolean

[JScript] public function IsEmpty() : Boolean;

Description

Returns whether or not this property bag contains all default values (is empty)

Returns whether or not this property bag contains all default values (is empty)

PageSetupDialog class (System.Windows.Forms)

a) ToString

Description

Represents a dialog box that allows users to manipulate page settings, including margins and paper orientation.

The **System.Windows.Forms.PageSetupDialog** dialog box modifies the

System.Drawing.Printing.PageSettings and

System.Drawing.Printing.PrinterSettings information for a given

System.Windows.Forms.PageSetupDialog.Document . The user can enable sections of the dialog box to manipulate printing, margins, and paper orientation, size, and source and to show help and network buttons.

b) *PageSetupDialog*

Example Syntax:

c) *ToString*

[C#]	public	PageSetupDialog();
[C++]	public:	PageSetupDialog();
[VB]	Public	Sub New()
[JScript]	public	function PageSetupDialog();

Description

Initializes a new instance of the **System.Windows.Forms.PageSetupDialog** class.

When an instance of **System.Windows.Forms.PageSetupDialog** is created, the following properties are initialized to the specified values.

d) *AllowMargins*

e) *ToString*

[C#]	public	bool	AllowMargins	{get; set;}
[C++]	public:	__property	bool	get_AllowMargins();public: __property void set_AllowMargins(bool);
[VB]	Public	Property	AllowMargins	As Boolean
[JScript]	public	function	get AllowMargins()	: Boolean;public function set AllowMargins(Boolean);

Description

Gets or sets a value indicating whether the margins section of the dialog box is enabled.

f) AllowOrientation

g) ToString

[C#] public bool AllowOrientation {get; set;}

[C++] public: __property bool get_AllowOrientation();public: __property void set_AllowOrientation(bool);

[VB] Public Property AllowOrientation As Boolean

[JScript] public function get AllowOrientation() : Boolean;public function set AllowOrientation(Boolean);

Description

Gets or sets a value indicating whether the orientation section of the dialog box (landscape vs. portrait) is enabled.

h) AllowPaper

i) ToString

[C#] public bool AllowPaper {get; set;}

[C++] public: __property bool get_AllowPaper();public: __property void set_AllowPaper(bool);

[VB] Public Property AllowPaper As Boolean

[JScript] public function get AllowPaper() : Boolean;public function set AllowPaper(Boolean);

1
2 *Description*

3 Gets or sets a value indicating whether the paper section of the dialog box
4 (paper size and paper source) is enabled.

5 *j) AllowPrinter*

6 *k) ToString*

7
8 [C#] public bool AllowPrinter {get; set;}

9 [C++] public: __property bool get_AllowPrinter();public: __property void
10 set_AllowPrinter(bool);

11 [VB] Public Property AllowPrinter As Boolean

12 [JScript] public function get AllowPrinter() : Boolean;public function set
13 AllowPrinter(Boolean);

14
15 *Description*

16 Gets or sets a value indicating whether the Printer button is enabled.

17 *l) Container*

18 *m) DesignMode*

19 *n) Document*

20 *o) ToString*

21
22
23 *Description*

24 Gets or sets a value indicating the **System.Drawing.Printing.PrintDocument**
25 to get page settings from.

- p) *Events*
- q) *MinMargins*
- r) *ToString*

Description

Gets or sets a value indicating the minimum margins the user is allowed to select, in hundredths of an inch.

- s) *PageSettings*
- t) *ToString*

```
[C#]      public      PageSettings      PageSettings      {get;      set;}
[C++] public: __property PageSettings* get_PageSettings();public: __property
void      set_PageSettings(PageSettings*);
[VB]      Public      Property      PageSettings      As      PageSettings
[JavaScript] public function get PageSettings() : PageSettings;public function set
PageSettings(PageSettings);
```

Description

Gets or sets a value indicating the page settings to modify.

- u) *PrinterSettings*
- v) *ToString*

```
[C#]      public      PrinterSettings      PrinterSettings      {get;      set;}
[C++] public: __property PrinterSettings* get_PrinterSettings();public: __property
```



```

1 void set_PrinterSettings(PrinterSettings*);
2 [VB] Public Property PrinterSettings As PrinterSettings
3 [JScript] public function get PrinterSettings() : PrinterSettings;public function set
4 PrinterSettings(PrinterSettings);

```

Description

Gets or sets the printer settings the dialog box to modify when the user clicks the Printer button.

w) *ShowHelp*

x) *ToString*

```

11 [C#] public bool ShowHelp {get; set;}
12 [C++] public: __property bool get_ShowHelp();public: __property void
13 set_ShowHelp(bool);
14 [VB] Public Property ShowHelp As Boolean
15 [JScript] public function get ShowHelp() : Boolean;public function set
16 ShowHelp(Boolean);

```

Description

Gets or sets a value indicating whether the Help button is visible.

y) *ShowNetwork*

z) *ToString*

```

23 [C#] public bool ShowNetwork {get; set;}
24 [C++] public: __property bool get_ShowNetwork();public: __property void
25

```

1 set_ShowNetwork(bool);

2 [VB] Public Property ShowNetwork As Boolean

3 [JScript] public function get ShowNetwork() : Boolean;public function set

4 ShowNetwork(Boolean);

6 *Description*

7 Gets or sets a value indicating whether the Network button is visible.

8 *aa) Site*

9 *bb) Reset*

11 [C#] public override void Reset();

12 [C++] public: void Reset();

13 [VB] Overrides Public Sub Reset()

14 [JScript] public override function Reset();

16 *Description*

17 Resets all options to their default values.

18 *cc) RunDialog*

20 [C#] protected override bool RunDialog(IntPtr hwndOwner);

21 [C++] protected: bool RunDialog(IntPtr hwndOwner);

22 [VB] Overrides Protected Function RunDialog(ByVal hwndOwner As IntPtr) As
23 Boolean

24 [JScript] protected override function RunDialog(hwndOwner : IntPtr) : Boolean;

Description

PaintEventArgs class (System.Windows.Forms)

a) ToString

Description

Provides data for the **System.Windows.Forms.Control.Paint** event.

The **System.Windows.Forms.Control.Paint** event occurs when a control is redrawn. A **System.Windows.Forms.PaintEventArgs** specifies the **System.Windows.Forms.PaintEventArgs.Graphics** to use to paint the control and the **System.Windows.Forms.PaintEventArgs.ClipRectangle** in which to paint.

b) PaintEventArgs

Example Syntax:

c) ToString

```
[C#] public PaintEventArgs(Graphics graphics, Rectangle clipRect);
```

```
[C++] public: PaintEventArgs(Graphics* graphics, Rectangle clipRect);
```

```
[VB] Public Sub New(ByVal graphics As Graphics, ByVal clipRect As  
Rectangle)
```

```
[JScript] public function PaintEventArgs(graphics : Graphics, clipRect :  
Rectangle);
```

Description

Initializes a new instance of the **System.Windows.Forms.PaintEventArgs** class with the specified graphics and clipping rectangle. The

System.Drawing.Graphics object used to paint the item. The **System.Drawing.Rectangle** that represents the rectangle in which to paint.

d) *ClipRectangle*

e) *ToString*

[C#] public Rectangle ClipRectangle {get;}

[C++] public: __property Rectangle get_ClippingRectangle();

[VB] Public ReadOnly Property ClipRectangle As Rectangle

[JScript] public function get ClipRectangle() : Rectangle;

Description

Indicates the rectangle in which to paint. This property is read-only.

f) *Graphics*

g) *ToString*

[C#] public Graphics Graphics {get;}

[C++] public: __property Graphics* get_Graphics();

[VB] Public ReadOnly Property Graphics As Graphics

[JScript] public function get Graphics() : Graphics;

Description

Indicates the **System.Drawing.Graphics** object used to paint. This property is read-only.

h) Dispose

[C#] public void Dispose();
[C++] public: __sealed void Dispose();
[VB] NotOverridable Public Sub Dispose()
[JScript] public function Dispose(); Releases the resources used by the
System.Windows.Forms.PaintEventArgs .

Description

Releases all resources used by the **System.Windows.Forms.PaintEventArgs** .

Calling **System.Windows.Forms.PaintEventArgs.Dispose** allows the resources used by the **System.Windows.Forms.PaintEventArgs** . to be reallocated for other purposes. For more information about **System.Windows.Forms.PaintEventArgs.Dispose** , see .

i) Dispose

[C#] protected virtual void Dispose(bool disposing);
[C++] protected: virtual void Dispose(bool disposing);
[VB] Overridable Protected Sub Dispose(ByVal disposing As Boolean)
[JScript] protected function Dispose(disposing : Boolean);

Description

Releases the unmanaged resources used by the **System.Windows.Forms.PaintEventArgs** and optionally releases the managed resources.

This method is called by the public method and the **System.Object.Finalize** method. **true** to release both managed and unmanaged resources; **false** to release only unmanaged resources.

j) *Finalize*

[C#]				~PaintEventArgs();
[C++]				~PaintEventArgs();
[VB]	Overrides	Protected	Sub	Finalize()
[JScript]	protected	override	function	Finalize();

Description

PaintEventHandler delegate (System.Windows.Forms)

a) *ToString*

Description

Represents the method that will handle the **System.Windows.Forms.Control.Paint** event of a **System.Windows.Forms.Control** class. The source of the event. A **System.Windows.Forms.PaintEventArgs** that contains the event data.

When you create a **System.Windows.Forms.PaintEventHandler** delegate, you identify the method that will handle the event. To associate the event with your event handler, add an instance of the delegate to the event. The event handler is called whenever the event occurs, unless you remove the delegate. For more information about handling events with delegates, see .

Panel class (System.Windows.Forms)

a) *ToString*

Description

Represents a Windows **System.Windows.Forms.Panel** control.

A **System.Windows.Forms.Panel** is a control that contains other controls. You can use a **System.Windows.Forms.Panel** to group collections of controls such as a group of **System.Windows.Forms.RadioButton** controls. As with other container controls such as the **System.Windows.Forms.GroupBox** control, if the **System.Windows.Forms.Panel** control's **System.Windows.Forms.Control.Enabled** property is set to **false**, the controls contained within the **System.Windows.Forms.Panel** will also be disabled.

b) Panel

Example Syntax:

c) ToString

[C#]	public	Panel();
[C++]	public:	Panel();
[VB]	Public Sub	New()
[JScript]	public function	Panel();

Description

Initializes a new instance of the **System.Windows.Forms.Panel** class.

- d) *AccessibilityObject*
- e) *AccessibleDefaultActionDescription*
- f) *AccessibleDescription*
- g) *AccessibleName*
- h) *AccessibleRole*
- i) *AllowDrop*
- j) *Anchor*
- k) *AutoScroll*
- l) *AutoScrollMargin*
- m) *AutoScrollMinSize*
- n) *AutoScrollPosition*
- o) *BackColor*
- p) *BackgroundImage*
- q) *BindingContext*
- r) *BorderStyle*
- s) *ToString*

Description

Indicates the border style for the control.

By default, the **System.Windows.Forms.Panel** control is displayed without a border. You can use this property to distinguish the boundaries of the **System.Windows.Forms.Panel** control from other areas on the form.

- t) *Bottom*
- u) *Bounds*
- v) *CanFocus*
- w) *CanSelect*
- x) *Capture*
- y) *CausesValidation*
- z) *ClientRectangle*
- aa) *ClientSize*
- bb) *CompanyName*
- cc) *Container*
- dd) *ContainsFocus*
- ee) *ContextMenu*
- ff) *Controls*
- gg) *Created*
- hh) *CreateParams*
- ii) *ToString*

Description

Returns the parameters needed to create the handle. Inheriting classes can override this to provide extra functionality. They should not, however, forget to call `base.CreateParams()` first to get the struct filled up with the basic info.

- 1 *jj) Cursor*
- 2 *kk) DataBindings*
- 3 *ll) DefaultImeMode*
- 4 *mm) DefaultSize*
- 5 *nn) ToString*

6

7

8 *Description*

9 Deriving classes can override this to configure a default size for their control.
10 This is more efficient than setting the size in the control's constructor.

MSI-861US.APP
509-324-9256
lee@hayes.ptc

1	oo) <i>DesignMode</i>
2	pp) <i>DisplayRectangle</i>
3	qq) <i>Disposing</i>
4	rr) <i>Dock</i>
5	ss) <i>DockPadding</i>
6	tt) <i>Enabled</i>
7	uu) <i>Events</i>
8	vv) <i>Focused</i>
9	ww) <i>Font</i>
10	xx) <i>FontHeight</i>
11	yy) <i>ForeColor</i>
12	zz) <i>Handle</i>
13	aaa) <i>HasChildren</i>
14	bbb) <i>Height</i>
15	ccc) <i>HScroll</i>
16	ddd) <i>ImeMode</i>
17	eee) <i>InvokeRequired</i>
18	fff) <i>IsAccessible</i>
19	ggg) <i>IsDisposed</i>
20	hhh) <i>IsHandleCreated</i>
21	iii) <i>Left</i>
22	jjj) <i>Location</i>
23	kkk) <i>Name</i>

1 *lll) Parent*

2 *mmm) ProductName*

3 *nnn) ProductVersion*

4 *ooo) RecreatingHandle*

5 *ppp) Region*

6 *qqq) RenderRightToLeft*

7 *rrr) ResizeRedraw*

8 *sss) Right*

9 *ttt) RightToLeft*

10 *uuu) ShowFocusCues*

11 *vvv) ShowKeyboardCues*

12 *www) Site*

13 *xxx) Size*

14 *yyy) TabIndex*

15 *zzz) TabStop*

16 *aaaa) ToString*

17

18

19

20 *Description*

21

22

23

24

25

bbbb) Tag

cccc) Text

dddd) ToString

Description

eeee) Top

ffff) TopLevelControl

gggg) Visible

hhhh) VScroll

iiii) Width

jjjj) WindowTarget

kkkk) ToString

llll) ToString

mmmm)ToString

nnnn) OnResize

[C#] protected override void OnResize(EventArgs eventargs);

[C++] protected: void OnResize(EventArgs* eventargs);

[VB] Overrides Protected Sub OnResize(ByVal eventargs As EventArgs)

[JScript] protected override function OnResize(eventargs : EventArgs);

Description

Fires the event indicating that the panel has been resized. Inheriting controls should use this in favour of actually listening to the event, but should not forget to call `base.OnResize()` to ensure that the event is still fired for external listeners.

Event to send

o o o o o) ToString

[C#]	public	override	string	ToString();
[C++]	public:		String*	ToString();
[VB]	Overrides	Public	Function	ToString() As String
[JScript]	public	override	function	ToString() : String;

Description

Returns a string representation for this control.

Return Value: String Returns a string representation for this control.

PictureBox class (System.Windows.Forms)

a) WndProc

Description

Represents a Windows picture box control for displaying an image.

Typically the **System.Windows.Forms.PictureBox** is used to display graphics from a bitmap, icon, JPEG, GIF or other image file types.

b) PictureBox

Example Syntax:

c) WndProc

[C#]	public	PictureBox();
------	--------	---------------

[C++]	public:	PictureBox();
[VB]	Public Sub	New()
[JScript]	public function	PictureBox();

Description

Initializes a new instance of the **System.Windows.Forms.PictureBox** class.

The following table shows initial property values for an instance of **System.Windows.Forms.PictureBox** .

- d) *AccessibilityObject*
- e) *AccessibleDefaultActionDescription*
- f) *AccessibleDescription*
- g) *AccessibleName*
- h) *AccessibleRole*
- i) *AllowDrop*
- j) *WndProc*

Description

- k) *Anchor*
- l) *BackColor*
- m) *BackgroundImage*
- n) *BindingContext*
- o) *BorderStyle*
- p) *WndProc*

Description

Indicates the border style for the control.

You can specify this property at design time or run time.

- q) *Bottom*
- r) *Bounds*
- s) *CanFocus*
- t) *CanSelect*
- u) *Capture*
- v) *CausesValidation*
- w) *WndProc*

Description

.

- x) *ClientRectangle*
- y) *ClientSize*
- z) *CompanyName*
- aa) *Container*
- bb) *ContainsFocus*
- cc) *ContextMenu*
- dd) *Controls*
- ee) *Created*
- ff) *CreateParams*
- gg) *WndProc*

Description

Returns the parameters needed to create the handle. Inheriting classes can override this to provide extra functionality. They should not, however, forget to call `base.CreateParams()` first to get the struct filled up with the basic info.

- hh) *Cursor*
- ii) *DataBindings*
- jj) *DefaultImeMode*
- kk) *WndProc*

Description

Gets a value indicating the the mode for Input Method Editor (IME) for the **System.Windows.Forms.PictureBox** .

1 **ll) *DefaultSize***

2 **mm) *WndProc***

3
4 [C#] protected override Size DefaultSize {get;}

5 [C++] protected: __property virtual Size get_DefaultSize();

6 [VB] Overrides Protected ReadOnly Property DefaultSize As Size

7 [JScript] protected function get DefaultSize() : Size;

8
9 *Description*

10 Deriving classes can override this to configure a default size for their control.
This is more efficient than setting the size in the control's constructor.

11 **nn) *DesignMode***

12 **oo) *DisplayRectangle***

13 **pp) *Disposing***

14 **qq) *Dock***

15 **rr) *Enabled***

16 **ss) *Events***

17 **tt) *Focused***

18 **uu) *Font***

19 **vv) *WndProc***

20
21
22
23 *Description*

ww) *FontHeight*

xx) *ForeColor*

yy) *WndProc*

Description

zz) *Handle*

aaa) *HasChildren*

bbb) *Height*

ccc) *Image*

ddd) *WndProc*

Description

Indicates the image that the **System.Windows.Forms.PictureBox** displays.

The **System.Windows.Forms.PictureBox.Image** property is set to the **System.Drawing.Image** to display. You can do this either at design time or at run time.

eee) *ImeMode*

fff) *WndProc*

```
[C#]      public      new      ImeMode      ImeMode      {get;      set;}
```

```
[C++] public: __property ImeMode get_ImeMode();public: __property void  
set_ImeMode(ImeMode);
```

```
[VB]      Public      Property      ImeMode      As      ImeMode
```

1 [JScript] public function get ImeMode() : ImeMode;public function set
2 ImeMode(ImeMode);

4 *Description*

5 Gets or sets the Input Method Editor(IME) mode supported by this control.

1 **ggg) InvokeRequired**

2 **hhh) IsAccessible**

3 **iii) IsDisposed**

4 **jjj) IsHandleCreated**

5 **kkk) Left**

6 **lll) Location**

7 **mmm) Name**

8 **nnn) Parent**

9 **ooo) ProductName**

10 **ppp) ProductVersion**

11 **qqq) RecreatingHandle**

12 **rrr) Region**

13 **sss) RenderRightToLeft**

14 **ttt) ResizeRedraw**

15 **uuu) Right**

16 **vvv) RightToLeft**

17 **www) WndProc**

18
19
20
21 *Description*
22
23
24
25

xxx) *ShowFocusCues*

yyy) *ShowKeyboardCues*

zzz) *Site*

aaaa) *Size*

bbbb) *SizeMode*

cccc) *WndProc*

Description

Indicates how the image is displayed.

Valid values for this property are taken from the **System.Windows.Forms.PictureBoxSizeMode** enumeration. By default, in **PictureBoxSizeMode.Normal** mode, the **System.Drawing.Image** is placed in the upper left corner of the **System.Windows.Forms.PictureBox**, and any part of the image too big for the **System.Windows.Forms.PictureBox** is clipped. Using the **PictureBoxSizeMode.StretchImage** value causes the image to stretch to fit the **System.Windows.Forms.PictureBox**.

dddd) *TabIndex*

eeee) *WndProc*

[C#] public new int TabIndex {get; set;}

[C++] public: __property int get_TabIndex();public: __property void
set_TabIndex(int);

[VB] Public Property TabIndex As Integer

[JScript] public function get TabIndex() : int;public function set TabIndex(int);

Description

ffff) TabStop

gggg) WndProc

[C#] public new bool TabStop {get; set;}

[C++] public: __property bool get_TabStop();public: __property void
set_TabStop(bool);

[VB] Public Property TabStop As Boolean

[JScript] public function get TabStop() : Boolean;public function set
TabStop(Boolean);

Description

hhhh) Tag

iiii) Text

jjjj) WndProc

Description

1 *kkkk) Top*

2 *llll) TopLevelControl*

3 *mmmm)Visible*

4 *nnnn) Width*

5 *oooo) WindowTarget*

6 *pppp) WndProc*

7 *qqqq) WndProc*

10 *Description*

12 *rrrr) WndProc*

14 [C#] public new event KeyPressEventHandler KeyPress;

15 [C++] public: __event KeyPressEventHandler* KeyPress;

16 [VB] Shadows Public Event KeyPress As KeyPressEventHandler

18 *Description*

20 *ssss) WndProc*

22 [C#] public new event KeyEventHandler KeyUp;

23 [C++] public: __event KeyEventHandler* KeyUp;

24 [VB] Shadows Public Event KeyUp As KeyEventHandler

Description

tttt) WndProc

uuuu) WndProc

Description

Occurs when **System.Windows.Forms.PictureBox.SizeMode** changes.

For more information about handling events, see .

vvvv) Dispose

[C#] protected override void Dispose(bool disposing);

[C++] protected: void Dispose(bool disposing);

[VB] Overrides Protected Sub Dispose(ByVal disposing As Boolean)

[JScript] protected override function Dispose(disposing : Boolean);

Description

www)OnEnabledChanged

[C#] protected override void OnEnabledChanged(EventArgs e);

[C++] protected: void OnEnabledChanged(EventArgs* e);

[VB] Overrides Protected Sub OnEnabledChanged(ByVal e As EventArgs)

[JScript] protected override function OnEnabledChanged(e : EventArgs);

Description

xxxx) OnPaint

[C#] protected override void OnPaint(PaintEventArgs pe);

[C++] protected: void OnPaint(PaintEventArgs* pe);

[VB] Overrides Protected Sub OnPaint(ByVal pe As PaintEventArgs)

[JScript] protected override function OnPaint(pe : PaintEventArgs);

Description

Overridden onPaint to make sure that the image is painted correctly.

yyyy) OnParentChanged

[C#] protected override void OnParentChanged(EventArgs e);

[C++] protected: void OnParentChanged(EventArgs* e);

[VB] Overrides Protected Sub OnParentChanged(ByVal e As EventArgs)

[JScript] protected override function OnParentChanged(e : EventArgs);

Description

zzzz) OnResize

[C#] protected override void OnResize(EventArgs e);

[C++] protected: void OnResize(EventArgs* e);

1 [VB] Overrides Protected Sub OnResize(ByVal e As EventArgs)

2 [JScript] protected override function OnResize(e : EventArgs);

3
4 *Description*

5 OnResize override to invalidate entire control in Stretch mode OnResize override
6 to invalidate entire control in Stretch mode

7 *aaaaa) OnSizeModeChanged*

8 [C#] protected virtual void OnSizeModeChanged(EventArgs e);

9 [C++] protected: virtual void OnSizeModeChanged(EventArgs* e);

10 [VB] Overridable Protected Sub OnSizeModeChanged(ByVal e As EventArgs)

11 [JScript] protected function OnSizeModeChanged(e : EventArgs);

12
13 *Description*

14 Raises the **System.Windows.Forms.PictureBox.SizeModeChanged** event.

15 Raising an event invokes the event handler through a delegate. For more
16 information, see . An **System.EventArgs** that contains the event data.

17 *bbbbbb) OnVisibleChanged*

18
19 [C#] protected override void OnVisibleChanged(EventArgs e);

20 [C++] protected: void OnVisibleChanged(EventArgs* e);

21 [VB] Overrides Protected Sub OnVisibleChanged(ByVal e As EventArgs)

22 [JScript] protected override function OnVisibleChanged(e : EventArgs);

23
24 *Description*

25

cccc) SetBoundsCore

[C#] protected override void SetBoundsCore(int x, int y, int width, int height, BoundsSpecified specified);

[C++] protected: void SetBoundsCore(int x, int y, int width, int height, BoundsSpecified specified);

[VB] Overrides Protected Sub SetBoundsCore(ByVal x As Integer, ByVal y As Integer, ByVal width As Integer, ByVal height As Integer, ByVal specified As BoundsSpecified)

[JScript] protected override function SetBoundsCore(x : int, y : int, width : int, height : int, specified : BoundsSpecified);

Description

Overrides Control.setBoundsCore to enforce autoSize.

dddd) ToString

[C#] public override string ToString();

[C++] public: String* ToString();

[VB] Overrides Public Function ToString() As String

[JScript] public override function ToString() : String;

Description

Returns a string representation for this control.

Return Value: String Returns a string representation for this control.

PictureBoxSizeMode enumeration (System.Windows.Forms)

a) WndProc

Description

Specifies how an image is positioned within a **System.Windows.Forms.PictureBox** .

Use the members of this enumeration to set the value of the **System.Windows.Forms.PictureBox.SizeMode** property of the **System.Windows.Forms.PictureBox** .

b) WndProc

[C#] public const PictureBoxSizeMode AutoSize;

[C++] public: const PictureBoxSizeMode AutoSize;

[VB] Public Const AutoSize As PictureBoxSizeMode

[JScript] public var AutoSize : PictureBoxSizeMode;

Description

The **System.Windows.Forms.PictureBox** is sized equal to the size of the image that it contains.

c) WndProc

[C#] public const PictureBoxSizeMode CenterImage;

[C++] public: const PictureBoxSizeMode CenterImage;

[VB] Public Const CenterImage As PictureBoxSizeMode

[JScript] public var CenterImage : PictureBoxSizeMode;

Description

The image is displayed in the center if the **System.Windows.Forms.PictureBox** is larger than the image. If the image is larger than the **System.Windows.Forms.PictureBox**, the picture is placed in the center of the **System.Windows.Forms.PictureBox** and the outside edges are clipped.

d) *WndProc*

[C#]	public	const	PictureBoxSizeMode		Normal;
[C++]	public:	const	PictureBoxSizeMode		Normal;
[VB]	Public	Const	Normal	As	PictureBoxSizeMode
[JScript]	public	var	Normal	:	PictureBoxSizeMode;

Description

The image is placed in the upper-left corner of the **System.Windows.Forms.PictureBox**. The image is clipped if it is larger than the **System.Windows.Forms.PictureBox** it is contained in.

e) *WndProc*

[C#]	public	const	PictureBoxSizeMode		StretchImage;
[C++]	public:	const	PictureBoxSizeMode		StretchImage;
[VB]	Public	Const	StretchImage	As	PictureBoxSizeMode
[JScript]	public	var	StretchImage	:	PictureBoxSizeMode;

Description

The image within the **System.Windows.Forms.PictureBox** is stretched or shrunk to fit the size of the **System.Windows.Forms.PictureBox**.

PrintControllerWithStatusDialog class (System.Windows.Forms)

a) *ToString*

Description

Controls how a document is printed.

This class implements a **System.Drawing.Printing.PrintController** and adds a status dialog. A print controller specifies how a **System.Drawing.Printing.PrintDocument** is printed.

b) *PrintControllerWithStatusDialog*

Example Syntax:

c) *ToString*

[C#] public PrintControllerWithStatusDialog(PrintController
underlyingController);

[C++] public: PrintControllerWithStatusDialog(PrintController*
underlyingController);

[VB] Public Sub New(ByVal underlyingController As PrintController)

[JScript] public function PrintControllerWithStatusDialog(underlyingController :
PrintController); Initializes a new instance of the
System.Windows.Forms.PrintControllerWithStatusDialog class.

Description

Initializes a new instance of the **System.Windows.Forms.PrintControllerWithStatusDialog** class, wrapping the supplied **System.Drawing.Printing.PrintController** . A **System.Drawing.Printing.PrintController** to encapsulate.

d) *PrintControllerWithStatusDialog*

Example Syntax:

e) *ToString*

```
[C#]      public      PrintControllerWithStatusDialog(PrintController
underlyingController,      string      dialogTitle);

[C++]      public:      PrintControllerWithStatusDialog(PrintController*
underlyingController,      String*      dialogTitle);

[VB] Public Sub New(ByVal underlyingController As PrintController, ByVal
dialogTitle      As      String)

[JScript] public function PrintControllerWithStatusDialog(underlyingController :
PrintController,      dialogTitle      :      String);
```

Description

Initializes a new instance of the **System.Windows.Forms.PrintControllerWithStatusDialog** class, wrapping the supplied **System.Drawing.Printing.PrintController** and specifying a title for the dialog. A **System.Drawing.Printing.PrintController** to encapsulate. A **System.String** containing a title for the status dialog.

f) *OnEndPage*

```
[C#]      public      override      void      OnEndPage(PrintDocument      document,
PrintPageEventArgs      e);

[C++]      public:      void      OnEndPage(PrintDocument* document, PrintPageEventArgs*
e);

[VB] Overrides Public Sub OnEndPage(ByVal document As PrintDocument,
ByVal      e      As      PrintPageEventArgs)
```


1 [JScript] public override function OnEndPage(document : PrintDocument, e :
2 PrintPageEventArgs);

3
4 *Description*

5 Implements EndPage by delegating to the underlying controller.

6 **g) OnEndPrint**

7
8 [C#] public override void OnEndPrint(PrintDocument document, PrintEventArgs
9 e);

10 [C++] public: void OnEndPrint(PrintDocument* document, PrintEventArgs* e);

11 [VB] Overrides Public Sub OnEndPrint(ByVal document As PrintDocument,
12 ByVal e As PrintEventArgs)

13 [JScript] public override function OnEndPrint(document : PrintDocument, e :
14 PrintEventArgs);

15
16 *Description*

17 Implements EndPrint by delegating to the underlying controller.

18 **h) OnStartPage**

19
20 [C#] public override Graphics OnStartPage(PrintDocument document,
21 PrintPageEventArgs e);

22 [C++] public: Graphics* OnStartPage(PrintDocument* document,
23 PrintPageEventArgs* e);

24 [VB] Overrides Public Function OnStartPage(ByVal document As
25 PrintDocument, ByVal e As PrintPageEventArgs) As Graphics

1 [JScript] public override function OnStartPage(document : PrintDocument, e :
2 PrintPageEventArgs) : Graphics;

3
4 *Description*

5 Implements StartPage by delegating to the underlying controller.

6 *i) OnStartPrint*

7
8 [C#] public override void OnStartPrint(PrintDocument document, PrintEventArgs
9 e);

10 [C++] public: void OnStartPrint(PrintDocument* document, PrintEventArgs* e);

11 [VB] Overrides Public Sub OnStartPrint(ByVal document As PrintDocument,
12 ByVal e As PrintEventArgs)

13 [JScript] public override function OnStartPrint(document : PrintDocument, e :
14 PrintEventArgs);

15
16 *Description*

17 Implements StartPrint by delegating to the underlying controller.

18 PrintDialog class (System.Windows.Forms)

19 *a) ToString*

20
21
22 *Description*

23 Allows users to select a printer and choose which portions of the document to
24 print.
25

When you create an instance of **System.Windows.Forms.PrintDialog** , the read/write properties are set to initial values. For a list of these values, see the **System.Windows.Forms.PrintDialog.#ctor** constructor.

b) PrintDialog

Example Syntax:

c) ToString

[C#] public PrintDialog();

[C++] public: PrintDialog();

[VB] Public Sub New()

[JScript] public function PrintDialog();

Description

Initializes a new instance of the **System.Windows.Forms.PrintDialog** class.

When you create an instance of **System.Windows.Forms.PrintDialog** , the following read/write properties are set to initial values.

d) AllowPrintToFile

e) ToString

[C#] public bool AllowPrintToFile {get; set;}

[C++] public: __property bool get_AllowPrintToFile();public: __property void set_AllowPrintToFile(bool);

[VB] Public Property AllowPrintToFile As Boolean

[JScript] public function get AllowPrintToFile() : Boolean;public function set AllowPrintToFile(Boolean);

Description

Gets or sets a value indicating whether the Print to file check box is enabled.

f) AllowSelection

g) ToString

[C#] public bool AllowSelection {get; set;}

[C++] public: __property bool get_AllowSelection();public: __property void set_AllowSelection(bool);

[VB] Public Property AllowSelection As Boolean

[JScript] public function get AllowSelection() : Boolean;public function set AllowSelection(Boolean);

Description

Gets or sets a value indicating whether the From... To... Page option button is enabled.

h) AllowSomePages

i) ToString

[C#] public bool AllowSomePages {get; set;}

[C++] public: __property bool get_AllowSomePages();public: __property void set_AllowSomePages(bool);

[VB] Public Property AllowSomePages As Boolean

[JScript] public function get AllowSomePages() : Boolean;public function set AllowSomePages(Boolean);

Description

Gets or sets a value indicating whether the Pages option button is enabled.

j) *Container*

k) *DesignMode*

l) *Document*

m) *ToString*

Description

Gets or sets a value indicating the **System.Drawing.Printing.PrintDocument** used to obtain **System.Drawing.Printing.PrinterSettings** .

n) *Events*

o) *PrinterSettings*

p) *ToString*

Description

Gets or sets the **System.Drawing.Printing.PrinterSettings** the dialog box to modify.

q) *PrintToFile*

r) *ToString*

[C#] public bool PrintToFile {get; set;}

[C++] public: __property bool get_PrintToFile();public: __property void

1 set_PrintToFile(bool);

2 [VB] Public Property PrintToFile As Boolean

3 [JScript] public function get PrintToFile() : Boolean;public function set

4 PrintToFile(Boolean);

5
6 *Description*

7 Gets or sets a value indicating whether the Print to file check box is checked.

8 s) *ShowHelp*

9 t) *ToString*

10
11 [C#] public bool ShowHelp {get; set;}

12 [C++] public: __property bool get_ShowHelp();public: __property void

13 set_ShowHelp(bool);

14 [VB] Public Property ShowHelp As Boolean

15 [JScript] public function get ShowHelp() : Boolean;public function set

16 ShowHelp(Boolean);

17
18 *Description*

19 Gets or sets a value indicating whether the Help button is displayed.

20 u) *ShowNetwork*

21 v) *ToString*

22
23 [C#] public bool ShowNetwork {get; set;}

24 [C++] public: __property bool get_ShowNetwork();public: __property void

1 set_ShowNetwork(bool);

2 [VB] Public Property ShowNetwork As Boolean

3 [JScript] public function get ShowNetwork() : Boolean;public function set

4 ShowNetwork(Boolean);

6 *Description*

7 Gets or sets a value indicating whether the Network button is displayed.

8 w) *Site*

9 x) *Reset*

11 [C#] public override void Reset();

12 [C++] public: void Reset();

13 [VB] Overrides Public Sub Reset()

14 [JScript] public override function Reset();

16 *Description*

17 Resets all options, the last selected printer, and the page settings to their default values.

18 y) *RunDialog*

21 [C#] protected override bool RunDialog(IntPtr hwndOwner);

22 [C++] protected: bool RunDialog(IntPtr hwndOwner);

23 [VB] Overrides Protected Function RunDialog(ByVal hwndOwner As IntPtr) As
24 Boolean

25 [JScript] protected override function RunDialog(hwndOwner : IntPtr) : Boolean;

Specifies a print dialog box.

Return Value: **true** if the dialog was successfully run; otherwise, **false** . A value that represents the window handle of the owner window for the print dialog box.

PrintPreviewControl class (System.Windows.Forms)

a) *ToString*

The raw "preview" part of print previewing, without any dialogs or buttons. Most `PrintPreviewControl`'s are found on `PrintPreviewDialog`'s, but they don't have to be.

When you create an instance of **System.Windows.Forms.PrintPreviewControl**, some of the read/write properties are set to initial values. For a list of these values, see the **System.Windows.Forms.PrintPreviewControl.#ctor** constructor.

b) *PrintPreviewControl*

Example Syntax:

c) *ToString*

[C#]	public	PrintPreviewControl();
[C++]	public:	PrintPreviewControl();
[VB]	Public Sub	New()
[JScript]	public function	PrintPreviewControl();

Description

1 Initializes a new instance of the
System.Windows.Forms.PrintPreviewControl class.

2 When you create an instance of
System.Windows.Forms.PrintPreviewControl , the following read/write
3 properties are initialized.

- 4 d) *AccessibilityObject*
- 5 e) *AccessibleDefaultActionDescription*
- 6 f) *AccessibleDescription*
- 7 g) *AccessibleName*
- 8 h) *AccessibleRole*
- 9 i) *AllowDrop*
- 10 j) *Anchor*
- 11 k) *AutoZoom*
- 12 l) *ToString*

14
15
16 *Description*

17 Gets or sets a value indicating whether resizing the control or changing the
18 number of pages shown automatically adjusts the
System.Windows.Forms.PrintPreviewControl.Zoom property.

- m) *BackColor*
- n) *BackgroundImage*
- o) *BindingContext*
- p) *Bottom*
- q) *Bounds*
- r) *CanFocus*
- s) *CanSelect*
- t) *Capture*
- u) *CausesValidation*
- v) *ClientRectangle*
- w) *ClientSize*
- x) *Columns*
- y) *ToString*

Description

Gets or sets the number of pages displayed horizontally across the screen.

1 **z) *CompanyName***

2 **aa) *Container***

3 **bb) *ContainsFocus***

4 **cc) *ContextMenu***

5 **dd) *Controls***

6 **ee) *Created***

7 **ff) *CreateParams***

8 **gg) *ToString***

11 ***Description***

12 Gets the CreateParams used to create the window. If a subclass overrides this
13 function, it must call the base implementation.

- hh) Cursor*
- ii) DataBindings*
- jj) DefaultImeMode*
- kk) DefaultSize*
- ll) DesignMode*
- mm) DisplayRectangle*
- nn) Disposing*
- oo) Dock*
- pp) Document*
- qq) ToString*

Description

Gets or sets a value indicating the document to preview.

1	rr)	<i>Enabled</i>
2	ss)	<i>Events</i>
3	tt)	<i>Focused</i>
4	uu)	<i>Font</i>
5	vv)	<i>FontHeight</i>
6	ww)	<i>ForeColor</i>
7	xx)	<i>Handle</i>
8	yy)	<i>HasChildren</i>
9	zz)	<i>Height</i>
10	aaa)	<i>ImeMode</i>
11	bbb)	<i>InvokeRequired</i>
12	ccc)	<i>IsAccessible</i>
13	ddd)	<i>IsDisposed</i>
14	eee)	<i>IsHandleCreated</i>
15	fff)	<i>Left</i>
16	ggg)	<i>Location</i>
17	hhh)	<i>Name</i>
18	iii)	<i>Parent</i>
19	jjj)	<i>ProductName</i>
20	kkk)	<i>ProductVersion</i>
21	lll)	<i>RecreatingHandle</i>
22	mmm)	<i>Region</i>
23	nnn)	<i>RenderRightToLeft</i>
24		
25		

ooo) *ResizeRedraw*

ppp) *Right*

qqq) *RightToLeft*

rrr) *Rows*

sss) *ToString*

Description

Gets or sets the number of pages displayed vertically down the screen.

ttt) *ShowFocusCues*

uuu) *ShowKeyboardCues*

vvv) *Site*

www) *Size*

xxx) *StartPage*

yyy) *ToString*

Description

Gets or sets the page number of the upper left page.

1 *zzz) TabIndex*

2 *aaaa) TabStop*

3 *bbbb) Tag*

4 *cccc) Text*

5 *dddd) ToString*

6
7
8 *Description*

10 *eeee) Top*

11 *ffff) TopLevelControl*

12 *gggg) UseAntiAlias*

13 *hhhh) ToString*

16 *Description*

17 Gets or sets a value indicating whether to use anti-aliasing when displaying the
18 print preview.

19 Anti-aliasing, also known as gray scaling, uses several shades of gray around
20 areas of curves and diagonals in text to give the text a smoother appearance.

1 *iiii) Visible*

2 *jjjj) Width*

3 *kkkk) WindowTarget*

4 *llll) Zoom*

5 *mmmm) ToString*

6
7
8 *Description*

9 Gets or sets a value indicating how large the pages will appear.

10 *nnnn) ToString*

11
12
13 *Description*

14 Occurs when the start page changes.

15 For more information about handling events, see .

16 *oooo) InvalidatePreview*

17
18 [C#] public void InvalidatePreview();

19 [C++] public: void InvalidatePreview();

20 [VB] Public Sub InvalidatePreview()

21 [JScript] public function InvalidatePreview();

22
23 *Description*

24 Refreshes the preview of the document.

25 Call this method if the document appearance has changed.

pppp) OnPaint

[C#] protected override void OnPaint(PaintEventArgs pevent);
[C++] protected: void OnPaint(PaintEventArgs* pevent);
[VB] Overrides Protected Sub OnPaint(ByVal pevent As PaintEventArgs)
[JScript] protected override function OnPaint(pevent : PaintEventArgs);

Description

Paints the control.

qqqq) OnResize

[C#] protected override void OnResize(EventArgs eventargs);
[C++] protected: void OnResize(EventArgs* eventargs);
[VB] Overrides Protected Sub OnResize(ByVal eventargs As EventArgs)
[JScript] protected override function OnResize(eventargs : EventArgs);

Description

Invalidate the layout, if necessary.

rrrr) OnStartPageChanged

[C#] protected virtual void OnStartPageChanged(EventArgs e);
[C++] protected: virtual void OnStartPageChanged(EventArgs* e);
[VB] Overridable Protected Sub OnStartPageChanged(ByVal e As EventArgs)
[JScript] protected function OnStartPageChanged(e : EventArgs);

Description

Raises the **System.Windows.Forms.PrintPreviewControl.StartPageChanged** event.

Raising an event invokes the event handler through a delegate. For more information, see . An **System.EventArgs** that contains the event data.

ssss) ResetBackColor

[C#]	public	override	void	ResetBackColor();
[C++]	public:		void	ResetBackColor();
[VB]	Overrides	Public	Sub	ResetBackColor()
[JScript]	public	override	function	ResetBackColor();

Description

Resets the back color to the defaults for the PrintPreviewControl.

tttt) ResetForeColor

[C#]	public	override	void	ResetForeColor();
[C++]	public:		void	ResetForeColor();
[VB]	Overrides	Public	Sub	ResetForeColor()
[JScript]	public	override	function	ResetForeColor();

Description

Resets the back color to the defaults for the PrintPreviewControl.

uuuu) WndProc

[C#] protected override void WndProc(ref Message m);
[C++] protected: void WndProc(Message* m);
[VB] Overrides Protected Sub WndProc(ByRef m As Message)
[JScript] protected override function WndProc(m : Message);

PrintPreviewDialog class (System.Windows.Forms)

a) WndProc

Description

Represents a dialog box form that contains a
System.Windows.Forms.PrintPreviewControl .

When you create an instance of
System.Windows.Forms.PrintPreviewDialog , some of the read/write
properties are set to initial values. For a list of these, see the
System.Windows.Forms.PrintPreviewDialog.#ctor constructor.

b) PrintPreviewDialog

Example Syntax:

c) WndProc

[C#] public PrintPreviewDialog();
[C++] public: PrintPreviewDialog();
[VB] Public Sub New()
[JScript] public function PrintPreviewDialog();

Description

Initializes a new instance of the
System.Windows.Forms.PrintPreviewDialog class.

When you create an instance of
System.Windows.Forms.PrintPreviewDialog , the
System.Windows.Forms.PrintPreviewDialog.Document property is
initialized to **null** . You can change the value for this property at run time.

d) *AcceptButton*

e) *WndProc*

[C#] public new IButtonControl AcceptButton {get; set;}

[C++] public: __property IButtonControl* get_AcceptButton();public: __property

void set_AcceptButton(IButtonControl*);

[VB] Public Property AcceptButton As IButtonControl

[JScript] public function get AcceptButton() : IButtonControl;public function set

AcceptButton(IButtonControl);

Description

Gets or sets the button on the form that is clicked when the user presses the
ENTER key.

This property allows you to designate a default action to occur when the user
presses the ENTER key in your application. The button assigned to this property
must be an **System.Windows.Forms.IButtonControl** that is on the current
form or located within a container on the current form.

1 *f) AccessibilityObject*

2 *g) AccessibleDefaultActionDescription*

3 *h) AccessibleDescription*

4 *i) WndProc*

7 *Description*

8 Gets or sets the accessible description of the control.

9 *j) AccessibleName*

10 *k) WndProc*

12 [C#] public new string AccessibleName {get; set;}

13 [C++] public: __property String* get_AccessibleName();public: __property void
14 set_AccessibleName(String*);

15 [VB] Public Property AccessibleName As String

16 [JScript] public function get AccessibleName() : String;public function set
17 AccessibleName(String);

19 *Description*

20 Gets or sets the accessible name of the control.

21 *l) AccessibleRole*

22 *m) WndProc*

24 [C#] public new AccessibleRole AccessibleRole {get; set;}

```

1 [C++] public: __property AccessibleRole get _AccessibleRole();public: __property
2 void set _AccessibleRole(AccessibleRole);
3 [VB] Public Property AccessibleRole As AccessibleRole
4 [JScript] public function get AccessibleRole() : AccessibleRole;public function set
5 AccessibleRole(AccessibleRole);

```

Description

The accessible role of the control.

- n) ActiveControl*
- o) ActiveMdiChild*
- p) AllowDrop*
- q) WndProc*

Description

The AllowDrop property. If AllowDrop is set to true then this control will allow drag and drop operations and events to be used.

- r) AllowTransparency*
- s) Anchor*
- t) WndProc*

Description

The current value of the anchor property. The anchor property determines which edges of the control are anchored to the container's edges.

u) *AutoScale*

v) *WndProc*

[C#] public new bool AutoScale {get; set;}

[C++] public: __property bool get_AutoScale();public: __property void
set_AutoScale(bool);

[VB] Public Property AutoScale As Boolean

[JScript] public function get AutoScale() : Boolean;public function set
AutoScale(Boolean);

Description

Gets or sets a value indicating whether the form adjusts its size to fit the height of the font used on the form and scales its controls.

You can use this property to allow your form and its controls to automatically adjust based on changes in the font. This can be useful in applications where the font might increase or decrease based on the language specified for use by Windows.

w) *AutoScaleBaseSize*

x) *WndProc*

[C#] public override Size AutoScaleBaseSize {get; set;}

[C++] public: __property virtual Size get_AutoScaleBaseSize();public: __property
virtual void set_AutoScaleBaseSize(Size);

[VB] Overrides Public Property AutoScaleBaseSize As Size

[JScript] public function get AutoScaleBaseSize() : Size;public function set
AutoScaleBaseSize(Size);

Description

PrintPreviewDialog does not support AutoScaleBaseSize.

y) *AutoScroll*

z) *WndProc*

[C#] public override bool AutoScroll {get; set;}

[C++] public: __property virtual bool get_AutoScroll();public: __property virtual

void set_AutoScroll(bool);

[VB] Overrides Public Property AutoScroll As Boolean

[JScript] public function get AutoScroll() : Boolean;public function set

AutoScroll(Boolean);

Description

Gets or sets a value indicating whether the form implements autoscrolling.

aa) *AutoScrollMargin*

bb) *WndProc*

[C#] public new Size AutoScrollMargin {get; set;}

[C++] public: __property Size get_AutoScrollMargin();public: __property void

set_AutoScrollMargin(Size);

[VB] Public Property AutoScrollMargin As Size

[JScript] public function get AutoScrollMargin() : Size;public function set

AutoScrollMargin(Size);

Description

Gets or sets the size of the auto-scroll margin.

The auto-scroll-margin is used to determine the distance from the edges of the scrollable control. If the distance from the edge of a child control to parent-scrollable control is less than the value assigned to this property, the appropriate scroll bar is displayed.

cc) *AutoScrollMinSize*

dd) *WndProc*

[C#] public new Size AutoScrollMinSize {get; set;}

[C++] public: __property Size get_AutoScrollMinSize();public: __property void set_AutoScrollMinSize(Size);

[VB] Public Property AutoScrollMinSize As Size

[JScript] public function get AutoScrollMinSize() : Size;public function set AutoScrollMinSize(Size);

Description

Gets or sets the minimum size of the auto-scroll.

The **System.Windows.Forms.PrintPreviewDialog.AutoScrollMinSize** property is used to manage the screen size allocated to the automatic scroll bars.

1 *ee) AutoScrollPosition*

2 *ff) BackColor*

3 *gg) WndProc*

4
5
6 *Description*

7 The background color of this control. This is an ambient property and will
8 always return a non-null value.

9 *hh) BackgroundImage*

10 *ii) WndProc*

11 [C#] public override Image BackgroundImage {get; set;}

12 [C++] public: __property virtual Image* get_BackgroundImage();public:

13 __property virtual void set_BackgroundImage(Image*);

14 [VB] Overrides Public Property BackgroundImage As Image

15 [JScript] public function get BackgroundImage() : Image;public function set

16 BackgroundImage(Image);

17
18 *Description*

19 The background image of the control.

1 *jj) BindingContext*

2 *kk) Bottom*

3 *ll) Bounds*

4 *mm) CancelButton*

5 *nn) WndProc*

6
7
8 *Description*

9 Gets or sets the button control that will be clicked when the user presses the
10 ESC key.

11 *oo) CanFocus*

12 *pp) CanSelect*

13 *qq) Capture*

14 *rr) CausesValidation*

15 *ss) WndProc*

16
17
18 *Description*

19 Gets or sets a value indicating whether entering the control causes validation for
20 all controls that require validation.

1 **tt) ClientRectangle**

2 **uu) ClientSize**

3 **vv) CompanyName**

4 **ww) Container**

5 **xx) ContainsFocus**

6 **yy) ContextMenu**

7 **zz) WndProc**

8
9
10 *Description*

11 The contextMenu associated with this control. The contextMenu will be shown
12 when the user right clicks the mouse on the control.

13 **aaa) ControlBox**

14 **bbb) WndProc**

15
16 [C#] public new bool ControlBox {get; set;}

17 [C++] public: __property bool get_ControlBox();public: __property void
18 set_ControlBox(bool);

19 [VB] Public Property ControlBox As Boolean

20 [JScript] public function get ControlBox() : Boolean;public function set
21 ControlBox(Boolean);

22
23 *Description*

24 Gets or sets a value indicating whether a control box is displayed in the caption
25 bar of the form.

If the **System.Windows.Forms.PrintPreviewDialog.ControlBox** property is set to **true** , the control box is displayed in the upper-left corner of the caption bar. The control box is where the user can click to access the system menu.

ccc) Controls

ddd) Created

eee) CreateParams

fff) Cursor

ggg) WndProc

Description

Retrieves the cursor that will be displayed when the mouse is over this control.

hhh) DataBindings

iii) WndProc

[C#] public new ControlBindingsCollection DataBindings {get;}

[C++] public: __property ControlBindingsCollection* get_DataBindings();

[VB] Public ReadOnly Property DataBindings As ControlBindingsCollection

[JScript] public function get DataBindings() : ControlBindingsCollection;

Description

Gets the data bindings for the control.

Use the **System.Windows.Forms.PrintPreviewDialog.DataBindings** property to access the **System.Windows.Forms.ControlBindingsCollection** . By adding **System.Windows.Forms.Binding** objects to the collection, you can data-bind any property of a control to the property of an object.

1 *jjj) DefaultImeMode*

2 *kkk) DefaultSize*

3 *lll) DesignMode*

4 *mmm) DesktopBounds*

5 *nnn) DesktopLocation*

6 *ooo) DialogResult*

7 *ppp) DisplayRectangle*

8 *qqq) Disposing*

9 *rrr) Dock*

10 *sss) WndProc*

11
12
13 *Description*

14 The dock property. The dock property controls to which edge of the container
15 this control is docked to. For example, when docked to the top of the container,
16 the control will be displayed flush at the top of the container, extending the
17 length of the container.

17 *ttt) DockPadding*

18 *uuu) WndProc*

19
20 [C#] public new ScrollableControl.DockPaddingEdges DockPadding {get;}
21

22 [C++] public: __property ScrollableControl.DockPaddingEdges*
23 get_DockPadding();
24

25 [VB] Public ReadOnly Property DockPadding As
26 ScrollableControl.DockPaddingEdges

```

1 [JScript]      public      function      get      DockPadding()      :
2 ScrollableControl.DockPaddingEdges;
3

```

Description

vvv) Document

www) WndProc

```

9 [C#]      public      PrintDocument      Document      {get;      set;}
10 [C++] public: __property PrintDocument* get_Document();public: __property
11 void      set_Document(PrintDocument*);
12 [VB]      Public      Property      Document      As      PrintDocument
13 [JScript] public function get Document() : PrintDocument;public function set
14 Document(PrintDocument);
15

```

Description

Gets or sets the document to preview.

This property is equivalent to
System.Windows.Forms.PrintPreviewControl.Document .

xxx) Enabled

yyy) WndProc

```

22
23 [C#]      public      new      bool      Enabled      {get;      set;}
24 [C++] public: __property bool get_Enabled();public: __property void
25 set_Enabled(bool);

```

1 [VB] Public Property Enabled As Boolean

2 [JScript] public function get Enabled() : Boolean;public function set

3 Enabled(Boolean);

4
5 *Description*

6 Get or sets a value indicating whether the control is enabled.

7 **zzz) Events**

8 **aaaa) Focused**

9 **bbbb) Font**

10 **cccc) WndProc**

11
12
13 *Description*

14 Retrieves the current font for this control. This will be the font used by default
15 for painting and text in the control.

16 **dddd) FontHeight**

17 **eeee) ForeColor**

18 **ffff) WndProc**

19
20
21 *Description*

22 The foreground color of the control.

gggg) FormBorderStyle

hhhh) WndProc

```
[C#]    public    new    FormBorderStyle    FormBorderStyle    {get;    set;}
[C++]    public:    __property    FormBorderStyle    get_FormBorderStyle();public:
__property    void    set_FormBorderStyle(FormBorderStyle);
[VB]    Public    Property    FormBorderStyle    As    FormBorderStyle
[JScript]    public    function    get    FormBorderStyle()    :    FormBorderStyle;public
function    set    FormBorderStyle(FormBorderStyle);
```

Description

Gets or sets the border style of the form.

The border style of the form determines how the outer edge of the form appears. In addition to changing the border display for a form, certain border styles prevent the form from being sized. For example, the **System.Windows.Forms.FormBorderStyle.FixedDialog** border style changes the border of the form to that of a dialog box and prevents the form from being resized. The border style can also affect the size or availability of the caption bar section of a form.

iiii) Handle

jjjj) HasChildren

kkkk) Height

llll) HelpButton

mmmm) WndProc

Description

Gets or sets a value indicating whether a help button should be displayed in the caption box of the form.

When this property is set to **true**, a small button with a question mark appears in the caption bar to the left of the close button. You can use this button to display help for your application. You can create an event handler for the **System.Windows.Forms.Control.HelpRequested** event of the **System.Windows.Forms.Control** class to display help information to the user when the help button of the form is clicked.

nnnn) HScroll

oooo) Icon

pppp) WndProc

Description

Gets or sets the icon for the form.

A form's icon designates the picture that represents the form in the taskbar as well as the icon that is displayed for the control box of the form.

qqqq) ImeMode

rrrr) WndProc

```
[C#]      public      new      ImeMode      ImeMode      {get;      set;}
```

```
[C++] public: __property ImeMode get_ImeMode();public: __property void  
set_ImeMode(ImeMode);
```

```
[VB]      Public      Property      ImeMode      As      ImeMode
```

```
[JScript] public function get ImeMode() : ImeMode;public function set  
ImeMode(ImeMode);
```

Description

Gets or sets the Input Method Editor(IME) mode supported by this control.

ssss) InvokeRequired

tttt) IsAccessible

uuuu) IsDisposed

vvvv) IsHandleCreated

wwwv) IsMdiChild

xxxx) IsMdiContainer

yyyy) WndProc

Description

Gets or sets a value indicating whether the form is a container for multiple document interface (MDI) child forms.

This property changes the display and behavior of the form to an MDI parent form. When this property is set to **true** , the form displays a sunken client area with a raised border. All MDI child forms assigned to the parent form are displayed within its client area.

zzzz) IsRestrictedWindow

aaaaa) KeyPreview

bbbbb) WndProc

Description

Gets or sets a value indicating whether the form will receive key events before the event is passed to the control that has focus.

When this property is set to true, the form will receive all **System.Windows.Forms.Control.KeyPress** ,

System.Windows.Forms.Control.KeyDown , and **System.Windows.Forms.Control.KeyUp** events. After the form's event handlers have completed processing the keystroke, the keystroke is then assigned to the control with focus.

cccc) Left

dddd) Location

eeee) WndProc

Description

Gets or sets the coordinates of the upper-left corner of the control relative to the upper-left corner of its container.

ffff) MaximizeBox

gggg) WndProc

[C#] public new bool MaximizeBox {get; set;}

[C++] public: __property bool get_MaximizeBox();public: __property void
set_MaximizeBox(bool);

[VB] Public Property MaximizeBox As Boolean

[JScript] public function get MaximizeBox() : Boolean;public function set
MaximizeBox(Boolean);

Description

Gets or sets a value indicating whether the maximize button is displayed in the caption bar of the form.

1 *hhhhh)MaximizedBounds*

2 *iiii) MaximumSize*

3 *jjjj) WndProc*

4
5
6 *Description*

7 Gets or sets the maximum size the form can be resized to.

8 This property enables you to limit the size of a form to a specified maximum size.
9 You can use this feature when displaying multiple windows at the same time, to
10 ensure that a single window does not cause other windows to be hidden.

11 *kkkkk) MdiChildren*

12 *llll) MdiParent*

13 *mmmmm)Menu*

14 *nnnnn)WndProc*

15
16 *Description*

17 Gets or sets the **System.Windows.Forms.MainMenu** that is displayed in the
18 form.

19 You can use this property to switch between complete menu sets at run time.
20 For example, you can define one **System.Windows.Forms.MainMenu** to be
21 displayed when your multiple document interface (MDI) form has no active MDI
22 child forms and another **System.Windows.Forms.MainMenu** to display when
23 a child window is displayed. You can also use a different
24 **System.Windows.Forms.MainMenu** when specific conditions exist in your
25 application that require displaying a different menu set.

ooooo) *MergedMenu*

ppppp) *MinimizeBox*

qqqqq) *WndProc*

Description

Gets or sets a value indicating whether the minimize button is displayed in the caption bar of the form.

A minimize button enables users to minimize a window to an icon. To display a minimize button, you must also set the form's `FormBorderStyle` property to either **`System.Windows.Forms.FormBorderStyle.FixedSingle`** , **`System.Windows.Forms.FormBorderStyle.Sizable`** , **`System.Windows.Forms.FormBorderStyle.Fixed3D`** , or **`System.Windows.Forms.FormBorderStyle.FixedDialog`** .

rrrrr) *MinimumSize*

sssss) *WndProc*

```
[C#]      public      new      Size      MinimumSize      {get;      set;}
```

```
[C++] public: __property Size get_MinimumSize();public: __property void  
set_MinimumSize(Size);
```

```
[VB]      Public      Property      MinimumSize      As      Size
```

```
[JScript] public function get MinimumSize() : Size;public function set  
MinimumSize(Size);
```

Description

Gets the minimum size the form can be resized to.

This property enables you to limit the size of a form to a specified minimum size. You can use this feature to prevent a user from sizing a window to an undesirable size.

1 *ttttt) Modal*

2 *uuuuuu)Name*

3 *vvvvv) Opacity*

4 *wwwww)WndProc*

6
7 *Description*

8 Opacity does not apply to PrintPreviewDialogs.

9 *xxxxxx) OwnedForms*

10 *yyyyyy) Owner*

11 *zzzzz) Parent*

12 *aaaaaaa)ParentForm*

13 *bbbbbb)PrintPreviewControl*

14 *cccccc)WndProc*

16
17 *Description*

18 Gets a value indicating the **System.Windows.Forms.PrintPreviewControl**
19 contained in this form.

dddddd)ProductName

eeeeee)ProductVersion

ffffff) RecreatingHandle

gggggg)Region

hhhhhh)RenderRightToLeft

iiiiii) ResizeRedraw

jjjjjj) Right

kkkkkk)RightToLeft

llllll) WndProc

Description

This is used for international applications where the language is written from RightToLeft. When this property is true, control placement and text will be from right to left.

mmmmmm)ShowFocusCues

nnnnnn)ShowInTaskbar

oooooo)WndProc

Description

Gets or sets a value indicating whether the form is displayed in the Windows taskbar.

If a form is parented within another form, the parented form is not displayed in the Windows taskbar.

ssssss) WndProc

Gets or sets the size of the form.

This property allows you to set both the height and width of the form at the same time instead of setting the **System.Windows.Forms.Control.Height** and **System.Windows.Forms.Control.Width** properties individually. If you want to set the size and location of a form, you can use the **System.Windows.Forms.Form.DesktopBounds** property to size and locate the form based on desktop coordinates or use the **System.Windows.Forms.Control.Bounds** property of the **System.Windows.Forms.Control** class to set the size and location of the form based on screen coordinates.

uuuuuu)WndProc

```
SizeGripStyle(SizeGripStyle);
```

Description

Parameter	1990-1991		1991-1992		1992-1993		1993-1994		1994-1995		1995-1996		1996-1997		1997-1998		1998-1999		1999-2000		2000-2001		2001-2002		2002-2003		2003-2004		2004-2005		2005-2006		2006-2007		2007-2008		2008-2009		2009-2010		2010-2011		2011-2012		2012-2013		2013-2014		2014-2015		2015-2016		2016-2017		2017-2018		2018-2019		2019-2020		2020-2021		2021-2022		2022-2023		2023-2024		2024-2025		2025-2026		2026-2027		2027-2028		2028-2029		2029-2030		2030-2031		2031-2032		2032-2033		2033-2034		2034-2035		2035-2036		2036-2037		2037-2038		2038-2039		2039-2040		2040-2041		2041-2042		2042-2043		2043-2044		2044-2045		2045-2046		2046-2047		2047-2048		2048-2049		2049-2050		2050-2051		2051-2052		2052-2053		2053-2054		2054-2055		2055-2056		2056-2057		2057-2058		2058-2059		2059-2060		2060-2061		2061-2062		2062-2063		2063-2064		2064-2065		2065-2066		2066-2067		2067-2068		2068-2069		2069-2070		2070-2071		2071-2072		2072-2073		2073-2074		2074-2075		2075-2076		2076-2077		2077-2078		2078-2079		2079-2080		2080-2081		2081-2082		2082-2083		2083-2084		2084-2085		2085-2086		2086-2087		2087-2088		2088-2089		2089-2090		2090-2091		2091-2092		2092-2093		2093-2094		2094-2095		2095-2096		2096-2097		2097-2098		2098-2099		2099-2100		2100-2101		2101-2102		2102-2103		2103-2104		2104-2105		2105-2106		2106-2107		2107-2108		2108-2109		2109-2110		2110-2111		2111-2112		2112-2113		2113-2114		2114-2115		2115-2116		2116-2117		2117-2118		2118-2119		2119-2120		2120-2121		2121-2122		2122-2123		2123-2124		2124-2125		2125-2126		2126-2127		2127-2128		2128-2129		2129-2130		2130-2131		2131-2132		2132-2133		2133-2134		2134-2135		2135-2136		2136-2137		2137-2138		2138-2139		2139-2140		2140-2141		2141-2142		2142-2143		2143-2144		2144-2145		2145-2146		2146-2147		2147-2148		2148-2149		2149-2150		2150-2151		2151-2152		2152-2153		2153-2154		2154-2155		2155-2156		2156-2157		2157-2158		2158-2159		2159-2160		2160-2161		2161-2162		2162-2163		2163-2164		2164-2165		2165-2166		2166-2167		2167-2168		2168-2169		2169-2170		2170-2171		2171-2172		2172-2173		2173-2174		2174-2175		2175-2176		2176-2177		2177-2178		2178-2179		2179-2180		2180-2181		2181-2182		2182-2183		2183-2184		2184-2185		2185-2186		2186-2187		2187-2188		2188-2189		2189-2190		2190-2191		2191-2192		2192-2193		2193-2194		2194-2195		2195-2196		2196-2197		2197-2198		2198-2199		2199-2200		2200-2201		2201-2202		2202-2203		2203-2204		2204-2205		2205-2206		2206-2207		2207-2208		2208-2209		2209-2210		2210-2211		2211-2212		2212-2213		2213-2214		2214-2215		2215-2216		2	
-----------	-----------	--	-----------	--	-----------	--	-----------	--	-----------	--	-----------	--	-----------	--	-----------	--	-----------	--	-----------	--	-----------	--	-----------	--	-----------	--	-----------	--	-----------	--	-----------	--	-----------	--	-----------	--	-----------	--	-----------	--	-----------	--	-----------	--	-----------	--	-----------	--	-----------	--	-----------	--	-----------	--	-----------	--	-----------	--	-----------	--	-----------	--	-----------	--	-----------	--	-----------	--	-----------	--	-----------	--	-----------	--	-----------	--	-----------	--	-----------	--	-----------	--	-----------	--	-----------	--	-----------	--	-----------	--	-----------	--	-----------	--	-----------	--	-----------	--	-----------	--	-----------	--	-----------	--	-----------	--	-----------	--	-----------	--	-----------	--	-----------	--	-----------	--	-----------	--	-----------	--	-----------	--	-----------	--	-----------	--	-----------	--	-----------	--	-----------	--	-----------	--	-----------	--	-----------	--	-----------	--	-----------	--	-----------	--	-----------	--	-----------	--	-----------	--	-----------	--	-----------	--	-----------	--	-----------	--	-----------	--	-----------	--	-----------	--	-----------	--	-----------	--	-----------	--	-----------	--	-----------	--	-----------	--	-----------	--	-----------	--	-----------	--	-----------	--	-----------	--	-----------	--	-----------	--	-----------	--	-----------	--	-----------	--	-----------	--	-----------	--	-----------	--	-----------	--	-----------	--	-----------	--	-----------	--	-----------	--	-----------	--	-----------	--	-----------	--	-----------	--	-----------	--	-----------	--	-----------	--	-----------	--	-----------	--	-----------	--	-----------	--	-----------	--	-----------	--	-----------	--	-----------	--	-----------	--	-----------	--	-----------	--	-----------	--	-----------	--	-----------	--	-----------	--	-----------	--	-----------	--	-----------	--	-----------	--	-----------	--	-----------	--	-----------	--	-----------	--	-----------	--	-----------	--	-----------	--	-----------	--	-----------	--	-----------	--	-----------	--	-----------	--	-----------	--	-----------	--	-----------	--	-----------	--	-----------	--	-----------	--	-----------	--	-----------	--	-----------	--	-----------	--	-----------	--	-----------	--	-----------	--	-----------	--	-----------	--	-----------	--	-----------	--	-----------	--	-----------	--	-----------	--	-----------	--	-----------	--	-----------	--	-----------	--	-----------	--	-----------	--	-----------	--	-----------	--	-----------	--	-----------	--	-----------	--	-----------	--	-----------	--	-----------	--	-----------	--	-----------	--	-----------	--	-----------	--	-----------	--	-----------	--	-----------	--	-----------	--	-----------	--	-----------	--	-----------	--	-----------	--	-----------	--	-----------	--	-----------	--	-----------	--	-----------	--	-----------	--	-----------	--	-----------	--	-----------	--	-----------	--	-----------	--	-----------	--	-----------	--	-----------	--	-----------	--	-----------	--	-----------	--	-----------	--	-----------	--	-----------	--	-----------	--	-----------	--	-----------	--	-----------	--	-----------	--	-----------	--	-----------	--	-----------	--	-----------	--	-----------	--	-----------	--	-----------	--	-----------	--	-----------	--	-----------	--	-----------	--	---	--

[C#]	public	new	object	Tag	{get;	set;}
[C++]	public:	__property	Object*	get_Tag();	public:	__property void set_Tag(Object*);
[VB]	Public	Property	Tag	As	Object	
[JScript]	public	function	get Tag()	: Object;	public	function set Tag(Object);

Gets or sets the object that contains data about the control.

Retrieves or assigns the data currently associated with the tree node. Any **System.Object** derived type can be assigned to this property. If this property is being set through the Windows Forms designer, only text can be assigned.

ddddddd)WndProc

[C#]	public	override	string	Text	{get;	set;}
[C++]	public:	__property	virtual	String*	get_Text();	public: __property virtual void set_Text(String*);
[VB]	Overrides	Public	Property	Text	As	String
[JScript]	public	function	get	Text()	: String;	public function set Text(String);

The current text associated with this control.

1 *eeeeeee)Top*

2 *ffffff) TopLevel*

3 *ggggggg)TopLevelControl*

4 *hhhhhhh)TopMost*

5 *iiiiiii) WndProc*

6
7
8 *Description*

9 Gets or sets a value indicating whether the form should be displayed as the top-most form of your application.

10 A top-most form is a form that overlaps all the other forms even if it is not the
11 active or foreground form. Top-most forms are always displayed at the highest
12 point in the Z-order of an application. You can use this method to create a form
13 that is always displayed in your application, such as a Find and Replace tool
14 window.

14 *jjjjjjj) TransparencyKey*

15 *kkkkkkk)WndProc*

16
17 [C#] public new Color TransparencyKey {get; set;}

18 [C++] public: __property Color get_TransparencyKey();public: __property void
19 set_TransparencyKey(Color);

20 [VB] Public Property TransparencyKey As Color

21 [JScript] public function get TransparencyKey() : Color;public function set
22 TransparencyKey(Color);

23
24 *Description*

25 Gets or sets the color that will represent transparent areas of the form.

When the **System.Windows.Forms.Form.TransparencyKey** property is assigned a **System.Drawing.Color**, the areas of the form that have the same **System.Windows.Forms.Control.BackColor** will be displayed transparently. Any mouse actions, such as the click of the mouse, that are performed on the transparent areas of the form will be transferred to the windows below the transparent area. For example, if the client region of a form is made transparent, clicking the mouse on that area would send the event notification of the click to any window that is below it. If the color assigned to the **System.Windows.Forms.Form.TransparencyKey** property is the same as any controls on the form, they also will be displayed transparently. For example, if you have a **System.Windows.Forms.Button** control on a form that has its **System.Windows.Forms.Form.TransparencyKey** property set to **SystemColors.Control**, the control will be displayed transparently unless the **System.Windows.Forms.Control.BackColor** property of the **System.Windows.Forms.Button** control is changed to a different color.

lllllll) UseAntiAlias

mmmmmmm)WndProc

[C#] public bool UseAntiAlias {get; set;}

[C++] public: __property bool get_UseAntiAlias();public: __property void
set_UseAntiAlias(bool);

[VB] Public Property UseAntiAlias As Boolean

[JScript] public function get UseAntiAlias() : Boolean;public function set
UseAntiAlias(Boolean);

Description

Gets or sets a value indicating whether to use anti-aliasing when displaying the print preview.

nnnnnnn)Visible

ooooooo)WndProc

```
[C#]      public      new      bool      Visible      {get;      set;}
[C++]    public:  __property  bool  get_Visible();public:  __property  void
set_Visible(bool);
[VB]      Public      Property      Visible      As      Boolean
[JScript] public  function  get  Visible()  :  Boolean;public  function  set
Visible(Boolean);
```

Description

Gets or sets a value indicating whether the control is visible.

ppppppp)VScroll

qqqqqqq)Width

rrrrrrr)WindowState

sssssss)WndProc

Description

Gets or sets the form's window state.

Before a form is displayed, the **System.Windows.Forms.Form.WindowState** property is always set to **FormWindowState.Normal** , regardless of its initial setting. This is reflected in the **System.Windows.Forms.Control.Height** , **System.Windows.Forms.Control.Left** , **System.Windows.Forms.Control.Top** , and **System.Windows.Forms.Control.Width** property settings. If a form is hidden after it has been shown, these properties reflect the previous state until the form

is shown again, regardless of any changes made to the **System.Windows.Forms.Form.WindowState** property.

tttttt) WindowTarget

uuuuuuu)CreateHandle

[C#]	protected	override	void	CreateHandle();
[C++]	protected:		void	CreateHandle();
[VB]	Overrides	Protected	Sub	CreateHandle()
[JScript]	protected	override	function	CreateHandle();

Description

Creates the handle for the form that encapsulates the **System.Windows.Forms.PrintPreviewDialog** .

For more information on **System.Windows.Forms.PrintPreviewDialog.CreateHandle** see the **System.Windows.Forms.Control.CreateHandle** method.

vvvvvvv)OnClosing

[C#]	protected	override	void	OnClosing(CancelEventArgs e);
[C++]	protected:		void	OnClosing(CancelEventArgs* e);
[VB]	Overrides	Protected	Sub	OnClosing(ByVal e As CancelEventArgs)
[JScript]	protected	override	function	OnClosing(e : CancelEventArgs);

Description

Forces the preview to be regenerated every time the dialog comes up Forces the preview to be regenerated every time the dialog comes up

ProgressBar class (System.Windows.Forms)

a) WndProc

Description

Represents a Windows progress bar control.

A **System.Windows.Forms.ProgressBar** control visually indicates the progress of a lengthy operation. The **System.Windows.Forms.ProgressBar** control displays a bar that fills in from left to right with the system highlight color as an operation progresses. The **System.Windows.Forms.ProgressBar** control is typically used when an application performs tasks such as a file copy operation or when printing a number of documents. Users of an application might consider an application unresponsive when a long task is in progress if there is no visual cue. By using the **System.Windows.Forms.ProgressBar** in your application, you alert the user that the application is performing a lengthy task and that the application is still responding.

b) ProgressBar

Example Syntax:

c) WndProc

[C#]	public	ProgressBar();
[C++]	public:	ProgressBar();
[VB]	Public	Sub New()
[JScript]	public	function ProgressBar();

Description

Initializes a new instance of the **System.Windows.Forms.ProgressBar** class.

By default, the **System.Windows.Forms.ProgressBar.Minimum** property is set to 0, the **System.Windows.Forms.ProgressBar.Maximum** property is

set to 100, and the **System.Windows.Forms.ProgressBar.Step** property is set to 10.

- d) *AccessibilityObject*
- e) *AccessibleDefaultActionDescription*
- f) *AccessibleDescription*
- g) *AccessibleName*
- h) *AccessibleRole*
- i) *AllowDrop*
- j) *WndProc*

Description

- k) *Anchor*
- l) *BackColor*
- m) *WndProc*

Description

Gets or sets the background color of the control.

- n) *BackgroundImage*
- o) *WndProc*

```
[C#] public override Image BackgroundImage {get; set;}
```

```
[C++] public: __property virtual Image* get_BackgroundImage();public:
```

```

1  __property      virtual      void      set_BackgroundImage(Image*);
2  [VB]  Overrides  Public  Property  BackgroundImage  As  Image
3  [JScript] public function get BackgroundImage() : Image;public function set
4  BackgroundImage(Image);

```

Description

- p) **BindingContext**
- q) **Bottom**
- r) **Bounds**
- s) **CanFocus**
- t) **CanSelect**
- u) **Capture**
- v) **CausesValidation**
- w) **WndProc**

Description

1 *x) ClientRectangle*

2 *y) ClientSize*

3 *z) CompanyName*

4 *aa) Container*

5 *bb) ContainsFocus*

6 *cc) ContextMenu*

7 *dd) Controls*

8 *ee) Created*

9 *ff) CreateParams*

10 *gg) WndProc*

11
12
13 *Description*

14 This is called when creating a window. Inheriting classes can override this to add
15 extra functionality, but should not forget to first call `base.CreateParams()` to
16 make sure the control continues to work correctly.

17 *hh) Cursor*

18 *ii) DataBindings*

19 *jj) DefaultImeMode*

20 *kk) WndProc*

21
22
23 *Description*

24 Gets the default Input Method Editor (IME) mode supported by this control.

As implemented in the **System.Windows.Forms.ProgressBar** class, this property always returns the **System.Windows.Forms.ImeMode.Disable** value.

ll) DefaultSize

mm) WndProc

[C#] protected override Size DefaultSize {get;}

[C++] protected: __property virtual Size get_DefaultSize();

[VB] Overrides Protected ReadOnly Property DefaultSize As Size

[JScript] protected function get DefaultSize() : Size;

Description

Gets the default size of the control.

nn) DesignMode

oo) DisplayRectangle

pp) Disposing

qq) Dock

rr) Enabled

ss) Events

tt) Focused

uu) Font

vv) WndProc

Description

Gets or sets the font of text in the **System.Windows.Forms.ProgressBar** .

ww) FontHeight

xx) ForeColor

yy) WndProc

Description

Gets or sets the foreground color of the control.

zz) Handle

aaa) HasChildren

bbb) Height

ccc) ImeMode

ddd) WndProc

Description

Gets or sets the Input Method Editor (IME) mode supported by this control.

eee) *InvokeRequired*

fff) *IsAccessible*

ggg) *IsDisposed*

hhh) *IsHandleCreated*

iii) *Left*

jjj) *Location*

kkk) *Maximum*

lll) *WndProc*

Description

Gets or sets the maximum value of the range of the control.

This property specifies the upper limit of the **System.Windows.Forms.ProgressBar.Value** property. When the value of the **System.Windows.Forms.ProgressBar.Maximum** property is changed, the **System.Windows.Forms.ProgressBar** control is redrawn to reflect the new range of the control. When the value of the **System.Windows.Forms.ProgressBar.Value** property is equal to the value of the **System.Windows.Forms.ProgressBar.Maximum** property, the progress bar is completely filled.

mmm) *Minimum*

nnn) *WndProc*

[C#] public int Minimum {get; set;}

[C++] public: __property int get_Minimum();public: __property void
set_Minimum(int);

[VB] Public Property Minimum As Integer

1 [JScript] public function get Minimum() : int;public function set Minimum(int);

3 *Description*

4 Gets or sets the minimum value of the range of the control.

5 This property specifies the lower limit of the
6 **System.Windows.Forms.ProgressBar.Value** property. When the value of
7 the **System.Windows.Forms.ProgressBar.Minimum** property is changed,
8 the **System.Windows.Forms.ProgressBar** control is redrawn to reflect the
9 new range of the control. When the value of the
10 **System.Windows.Forms.ProgressBar.Value** property is equal to the value
11 of the **System.Windows.Forms.ProgressBar.Minimum** property, the
12 progress bar is empty. To change the value of the progress bar, use the
13 **System.Windows.Forms.ProgressBar.Step** property with the
14 **System.Windows.Forms.ProgressBar.PerformStep** method, use the
15 **System.Windows.Forms.ProgressBar.Increment(System.Int32)** method,
16 or set the value of the **System.Windows.Forms.ProgressBar.Value** property
17 directly.

1	<i>ooo) Name</i>
2	<i>ppp) Parent</i>
3	<i>qqq) ProductName</i>
4	<i>rrr) ProductVersion</i>
5	<i>sss) RecreatingHandle</i>
6	<i>ttt) Region</i>
7	<i>uuu) RenderRightToLeft</i>
8	<i>vvv) ResizeRedraw</i>
9	<i>www) Right</i>
10	<i>xxx) RightToLeft</i>
11	<i>yyy) WndProc</i>

Description

1 *zzz) ShowFocusCues*

2 *aaaa) ShowKeyboardCues*

3 *bbbb) Site*

4 *cccc) Size*

5 *dddd) Step*

6 *eeee) WndProc*

7
8
9 *Description*

10 Gets or sets the amount by which a call to the
11 **System.Windows.Forms.ProgressBar.PerformStep** method increases the
12 current position of the progress bar .

13 You can use the **System.Windows.Forms.ProgressBar.Step** property to
14 specify the amount that each completed task in an operation changes the value
15 of the progress bar. For example, if you are copying a group of files, you may
16 want to set the value of the **System.Windows.Forms.ProgressBar.Step**
17 property to 1 and the value of the
18 **System.Windows.Forms.ProgressBar.Maximum** property to the total
19 number of files to copy. When each file is copied, you can call the
20 **System.Windows.Forms.ProgressBar.PerformStep** method to increment
21 the progress bar by the value of the
22 **System.Windows.Forms.ProgressBar.Step** property. If you want to have
23 more flexible control of the value of the progress bar, you can use the
24 **System.Windows.Forms.ProgressBar.Increment(System.Int32)** method
25 or set the value of the **System.Windows.Forms.ProgressBar.Value** property
26 directly.

ffff) TabIndex

gggg) TabStop

hhhh) WndProc

Description

iiii) Tag

jjjj) Text

kkkk) WndProc

Description

lll) Top

mmmm)TopLevelControl

nnnn) Value

oooo) WndProc

Description

Gets or sets the current position of the progress bar.

The minimum and maximum values of the **System.Windows.Forms.ProgressBar.Value** property are specified by the **System.Windows.Forms.ProgressBar.Minimum** and **System.Windows.Forms.ProgressBar.Maximum** properties. This property enables you to increment or decrement the value of the progress bar directly. To perform consistent increases in the value of the **System.Windows.Forms.ProgressBar** control you can use the

System.Windows.Forms.ProgressBar.Step property with the **System.Windows.Forms.ProgressBar.PerformStep** method. To increase the progress bar value by varying amounts, use the **System.Windows.Forms.ProgressBar.Increment(System.Int32)** method.

pppp) Visible

qqqq) Width

rrrr) WindowTarget

ssss) WndProc

tttt) WndProc

uuuu) WndProc

vvvv) WndProc

wwwv) WndProc

xxxx) WndProc

yyyy) CreateHandle

[C#]	protected	override	void	CreateHandle();
[C++]	protected:		void	CreateHandle();
[VB]	Overrides	Protected	Sub	CreateHandle()
[JScript]	protected	override	function	CreateHandle();

Description

zzzz) Increment

[C#]	public	void	Increment(int	value);
[C++]	public:	void	Increment(int	value);

```

1 [VB]      Public      Sub      Increment(ByVal      value      As      Integer)
2 [JScript]      public      function      Increment(value      :      int);
3

```

Description

Advances the current position of the progress bar by the specified amount.

The **System.Windows.Forms.ProgressBar.Increment(System.Int32)** method enables you to increment the value of the progress bar by a specific amount. This method of incrementing the progress bar is similar to using the **System.Windows.Forms.ProgressBar.Step** property with the **System.Windows.Forms.ProgressBar.PerformStep** method. The **System.Windows.Forms.ProgressBar.Value** property specifies the current position of the **System.Windows.Forms.ProgressBar**. If, after calling the **System.Windows.Forms.ProgressBar.Increment(System.Int32)** method, the new value is greater than the value of the **System.Windows.Forms.ProgressBar.Maximum** property, the value is reset to the value of the **System.Windows.Forms.ProgressBar.Minimum** property. The specified amount by which to increment the progress bar's current position.

aaaaa) OnHandleCreated

```

15 [C#]      protected      override      void      OnHandleCreated(EventArgs      e);
16 [C++]      protected:      void      OnHandleCreated(EventArgs*      e);
17 [VB]      Overrides Protected Sub OnHandleCreated(ByVal e As EventArgs)
18 [JScript]      protected      override      function      OnHandleCreated(e      :      EventArgs);
19

```

Description

Overridden to set up our properties.

bbbbbb) PerformStep

```

24 [C#]      public      void      PerformStep();
25

```

[C++]	public:	void	PerformStep();
[VB]	Public	Sub	PerformStep()
[JScript]	public	function	PerformStep();

Description

Advances the current position of the progress bar by the amount of the **System.Windows.Forms.ProgressBar.Step** property.

The **System.Windows.Forms.ProgressBar.Increment(System.Int32)** method increments the value of the progress bar by the amount specified by the Step property. You can use the **System.Windows.Forms.ProgressBar.Step** property to specify the amount that each completed task in an operation changes the value of the progress bar. For example, if you are copying a group of files, you may want to set the value of the **System.Windows.Forms.ProgressBar.Step** property to 1 and the value of the **System.Windows.Forms.ProgressBar.Maximum** property to the total number of files to copy. When each file is copied, you can call the **System.Windows.Forms.ProgressBar.PerformStep** method to increment the progress bar by the value of the **System.Windows.Forms.ProgressBar.Step** property. If you want to have more flexible control of the value of the progress bar, you can use the **System.Windows.Forms.ProgressBar.Increment(System.Int32)** method or set the value of the **System.Windows.Forms.ProgressBar.Value** property directly.

ccccc) ToString

[C#]	public	override	string	ToString();
[C++]	public:		String*	ToString();
[VB]	Overrides	Public	Function	ToString() As String
[JScript]	public	override	function	ToString() : String;

Description

Returns a string representation for this control.

Return Value: String Returns a string representation for this control.

PropertyGrid class (System.Windows.Forms)

a) *WndProc*

Description

Provides a user interface for browsing the properties of an object.

To use the property grid, you create a new instance of the **System.Windows.Forms.PropertyGrid** class on a parent control and set **System.Windows.Forms.PropertyGrid.SelectedObject** to the object to display the properties for.

b) *PropertyGrid*

Example Syntax:

c) *WndProc*

[C#]	public	PropertyGrid();
[C++]	public:	PropertyGrid();
[VB]	Public	Sub New()
[JScript]	public	function PropertyGrid();

Description

Initializes a new instance of the **System.Windows.Forms.PropertyGrid** class.

To populate the grid, you must set the **System.Windows.Forms.PropertyGrid.SelectedObject** property.

- d) *AccessibilityObject*
- e) *AccessibleDefaultActionDescription*
- f) *AccessibleDescription*
- g) *AccessibleName*
- h) *AccessibleRole*
- i) *ActiveControl*
- j) *AllowDrop*
- k) *Anchor*
- l) *AutoScroll*
- m) *AutoScrollMargin*
- n) *AutoScrollMinSize*
- o) *AutoScrollPosition*
- p) *BackColor*
- q) *WndProc*

Description

r) *BackgroundImage*

s) *WndProc*

```
[C#] public override Image BackgroundImage {get; set;}
```

```
[C++] public: __property virtual Image* get_BackgroundImage();public:
```

```

1  __property          virtual          void          set_BackgroundImage(Image*);
2  [VB] Overrides     Public     Property     BackgroundImage     As     Image
3  [JScript] public function get BackgroundImage() : Image;public function set
4  BackgroundImage(Image);
5

```

Description

- t) *BindingContext*
- u) *Bottom*
- v) *Bounds*
- w) *BrowsableAttributes*
- x) *WndProc*

Description

Gets or sets the browsable attributes associated with the object that the property grid is attached to.

Assigning an attribute collection that is empty or is **null** causes a new **System.ComponentModel.AttributeCollection** to be created with **System.ComponentModel.BrowsableAttribute.Yes** set.

1 y) *CanFocus*

2 z) *CanSelect*

3 aa) *CanShowCommands*

4 bb) *WndProc*

5
6
7 *Description*

8 Gets a value indicating whether the commands pane can be made visible for the
9 currently selected objects.

10 This property is **true** if

11 **System.Windows.Forms.PropertyGrid.CommandsVisibleIfAvailable** is
12 **true** and the object has commands available.

13 cc) *Capture*

14 dd) *CausesValidation*

15 ee) *ClientRectangle*

16 ff) *ClientSize*

17 gg) *CommandsBackColor*

18 hh) *WndProc*

19
20 *Description*

21 Gets or sets the background color of the hot commands region.

ii) *CommandsForeColor*

jj) *WndProc*

```
[C#]      public      Color      CommandsForeColor      {get;      set;}
[C++] public: __property Color get_CommandsForeColor();public: __property
void      set_CommandsForeColor(Color);
[VB]      Public      Property      CommandsForeColor      As      Color
[JScript] public function get CommandsForeColor() : Color;public function set
CommandsForeColor(Color);
```

Description

Gets or sets the foreground color for the hot commands region.

kk) *CommandsVisible*

ll) *WndProc*

```
[C#]      public      virtual      bool      CommandsVisible      {get;}
[C++] public:      __property      virtual      bool      get_CommandsVisible();
[VB] Overridable Public ReadOnly Property CommandsVisible As Boolean
[JScript] public function get CommandsVisible() : Boolean;
```

Description

Gets a value indicating whether the commands pane is visible.

This property is true if the selected object has commands available and the **System.Windows.Forms.PropertyGrid.CommandsVisibleIfAvailable** property is **true** .

1 *mm) CommandsVisibleIfAvailable*

2 *nn) WndProc*

3
4 [C#] public virtual bool CommandsVisibleIfAvailable {get; set;}

5 [C++] public: __property virtual bool get_CommandsVisibleIfAvailable();public:

6 __property virtual void set_CommandsVisibleIfAvailable(bool);

7 [VB] Overridable Public Property CommandsVisibleIfAvailable As Boolean

8 [JScript] public function get CommandsVisibleIfAvailable() : Boolean;public

9 function set CommandsVisibleIfAvailable(Boolean);

10
11 *Description*

12 Gets a value indicating whether the commands pane is visible for objects that
13 expose verbs.

14 *oo) CompanyName*

15 *pp) Container*

16 *qq) ContainsFocus*

17 *rr) ContextMenu*

18 *ss) ContextMenuDefaultLocation*

19 *tt) WndProc*

20
21
22 *Description*

23 Gets the default location for the context menu.

24 The context menu location is the center of the active property label in the grid.
25 You can use this information to position the context menu when the menu is
invoked using the keyboard.

1 **uu) Controls**

2 **vv) WndProc**

3
4 [C#] public new Control.ControlCollection Controls {get;}

5 [C++] public: __property Control.ControlCollection* get_Controls();

6 [VB] Public ReadOnly Property Controls As Control.ControlCollection

7 [JScript] public function get Controls() : Control.ControlCollection;

8
9 *Description*

10 Collection of child controls.

11 **ww) Created**

12 **xx) CreateParams**

13 **yy) Cursor**

14 **zz) DataBindings**

15 **aaa) DefaultImeMode**

16 **bbb) DefaultSize**

17 **ccc) WndProc**

18
19
20 *Description*

ddd) *DefaultTabType*

eee) *WndProc*

```
[C#]      protected      virtual      Type      DefaultTabType      {get;}
[C++]     protected:     __property  virtual  Type*    get_DefaultTabType();
[VB]      Overridable Protected ReadOnly Property DefaultTabType As Type
[JScript] protected      function    get      DefaultTabType()      :      Type;
```

Description

Gets the type of the default tab.

The default implementation of this property returns
System.Windows.Forms.Design.PropertyTab .

fff) DesignMode
ggg) DisplayRectangle
hhh) Disposing
iii) Dock
jjj) DockPadding
kkk) DrawFlatToolbar
lll) WndProc
mmm) Enabled
nnn) Events
ooo) Focused
ppp) Font
qqq) FontHeight
rrr) ForeColor
sss) WndProc

Description

1 *ttt) Handle*

2 *uuu) HasChildren*

3 *vvv) Height*

4 *www) HelpBackColor*

5 *xxx) WndProc*

6
7
8 *Description*

9 Gets or sets the background color for the Help region.

10 *yyy) HelpForeColor*

11 *zzz) WndProc*

12
13 [C#] public Color HelpForeColor {get; set;}

14 [C++] public: __property Color get_HelpForeColor();public: __property void
15 set_HelpForeColor(Color);

16 [VB] Public Property HelpForeColor As Color

17 [JScript] public function get HelpForeColor() : Color;public function set
18 HelpForeColor(Color);

19
20 *Description*

21 Gets or sets the foreground color for the Help region.
22
23
24
25

aaaa) *HelpVisible*

bbbb) *WndProc*

```
[C#]      public      virtual      bool      HelpVisible      {get;      set;}
[C++] public: __property virtual bool get_HelpVisible();public: __property virtual
void                                             set_HelpVisible(bool);
[VB]      Overridable      Public      Property      HelpVisible      As      Boolean
[JScript] public function get HelpVisible() : Boolean;public function set
HelpVisible(Boolean);
```

Description

Gets or sets a value indicating whether the Help text is visible.

This method allows the display of any available help text associated with the selected property in the grid.

cccc) *HScroll*

dddd) *ImeMode*

eeee) *InvokeRequired*

ffff) *IsAccessible*

gggg) *IsDisposed*

hhhh) *IsHandleCreated*

iiii) *LargeButtons*

jjjj) *WndProc*

Description

Gets or sets a value indicating whether buttons appear in standard size or in large size.

You can use this property to enhance accessibility. Large buttons are 32 by 32 pixels rather than the standard 16 by 16 pixels.

kkkk) Left

llll) LineColor

mmmm)WndProc

Description

Gets or sets the color of the gridlines and borders.

nnnn) Location

oooo) Name

pppp) Parent

qqqq) ParentForm

rrrr) ProductName

ssss) ProductVersion

tttt) PropertySort

uuuu) WndProc

Description

Gets or sets the type of sorting the **System.Windows.Forms.PropertyGrid** uses to display properties.

When you set the **System.Windows.Forms.PropertyGrid.PropertySort** property, the appearance of the property sort buttons on the grid changes change to reflect the current state of the of the property.

vvvv) PropertyTabs

www)WndProc

[C#] public PropertyGrid.PropertyTabCollection PropertyTabs {get;}

[C++] public: __property PropertyGrid.PropertyTabCollection*
get_PropertyTabs();

[VB] Public ReadOnly Property PropertyTabs As
PropertyGrid.PropertyTabCollection

[JScript] public function get PropertyTabs() :
PropertyGrid.PropertyTabCollection;

Description

Gets a new collection of property tabs.

xxxx) *RecreatingHandle*

yyyy) *Region*

zzzz) *RenderRightToLeft*

aaaaa) *ResizeRedraw*

bbbbbb) *Right*

cccc) *RightToLeft*

dddd) *SelectedItem*

eeee) *WndProc*

Description

Gets or sets the selected grid item.

Each **System.Windows.Forms.GridItem** corresponds to a property of the **System.Windows.Forms.PropertyGrid.SelectedObject** .

ffff) *SelectedObject*

ggggg) *WndProc*

[C#] public object SelectedObject {get; set;}

[C++] public: __property Object* get_SelectedObject();public: __property void
set_SelectedObject(Object*);

[VB] Public Property SelectedObject As Object

[JScript] public function get SelectedObject() : Object;public function set
SelectedObject(Object);

Description

Gets or sets the object for which the grid displays properties.

The **System.Windows.Forms.PropertyGrid.SelectedObject** property sets a single object into the grid to be browsed. If multiple objects are being browsed, this property returns the first one in the list. If no objects are selected, **null** is returned.

hhhhh)SelectedObjects

iiii) WndProc

[C#] public object[] SelectedObjects {get; set;}

[C++] public: __property Object* get_SelectedObjects();public: __property void set_SelectedObjects(Object* __gc[]);

[VB] Public Property SelectedObjects As Object ()

[JScript] public function get SelectedObjects() : Object[];public function set SelectedObjects(Object[]);

Description

Gets or sets the currently selected objects.

The **System.Windows.Forms.PropertyGrid** only displays the properties that are common to all the objects that are in the array. Assigning an array to the **System.Windows.Forms.PropertyGrid.SelectedObjects** replaces the reference to any **System.Windows.Forms.PropertyGrid.SelectedObject** you might make.

jjjj) SelectedTab

kkkk) WndProc

[C#] public PropertyTab SelectedTab {get;}

[C++] public: __property PropertyTab* get_SelectedTab();

[VB] Public ReadOnly Property SelectedTab As PropertyTab

1 [JScript] public function get SelectedTab() : PropertyTab;

3 *Description*

4 Gets the currently selected tab.

5 *lllll) ShowFocusCues*

6 *mmmmm)WndProc*

8 [C#] protected override bool ShowFocusCues {get;}

9 [C++] protected: __property virtual bool get_ShowFocusCues();

10 [VB] Overrides Protected ReadOnly Property ShowFocusCues As Boolean

11 [JScript] protected function get ShowFocusCues() : Boolean;

13 *Description*

15 *nnnnn)ShowKeyboardCues*

16 *ooooo) Site*

17 *ppppp) WndProc*

20 *Description*

qqqqq) Size

rrrrr) TabIndex

sssss) TabStop

ttttt) Tag

uuuuu)Text

vvvvv) ToolbarVisible

wwwww)WndProc

Description

Gets or sets a value indicating whether the toolbar is visible.

xxxxx) Top

yyyyy) TopLevelControl

zzzzz) ViewBackColor

aaaaaa)WndProc

Description

Gets or sets a value indicating the background color in the grid.

bbbbbb)ViewForeColor

cccccc)WndProc

[C#] public Color ViewForeColor {get; set;}

[C++] public: __property Color get_ViewForeColor();public: __property void

1 set_ViewForeColor(Color);

2 [VB] Public Property ViewForeColor As Color

3 [JScript] public function get ViewForeColor() : Color;public function set

4 ViewForeColor(Color);

5
6 *Description*

7 Gets or sets a value indicating the color of the text in the grid.

8 *dddddd)Visible*

9 *eeeeee)VScroll*

10 *ffffff) Width*

11 *gggggg)WindowTarget*

12 *hhhhh)WndProc*

13
14
15 *Description*

16 Occurs when the sort mode is changed.

17 For more information about handling events, see .

18 *iiiiii) WndProc*

19
20 [C#] public event PropertyTabChangedEventHandler PropertyTabChanged;

21 [C++] public: __event PropertyTabChangedEventHandler* PropertyTabChanged;

22 [VB] Public Event PropertyTabChanged As PropertyTabChangedEventHandler

23
24 *Description*

Occurs when a property tab changes.

For more information about handling events, see .

jjjjj) WndProc

[C#] public event PropertyChangedEventHandler PropertyChanged;

[C++] public: __event PropertyChangedEventHandler*

PropertyChanged;

[VB] Public Event PropertyChanged As

PropertyChangedEventHandler

Description

Occurs when a property value changes.

For more information about handling events, see .

kkkkkk) WndProc

Description

Occurs when the selected **System.Windows.Forms.GridItem** is changed.

For more information about handling events, see .

lllll) WndProc

[C#] public event EventHandler SelectedObjectsChanged;

[C++] public: __event EventHandler* SelectedObjectsChanged;

[VB] Public Event SelectedObjectsChanged As EventHandler

Description

Occurs when the objects selected by the **System.Windows.Forms.PropertyGrid.SelectedObjects** property have changed.

For more information about handling events, see .

CollapseAllGridItems

[C#]	public	void	CollapseAllGridItems();
[C++]	public:	void	CollapseAllGridItems();
[VB]	Public	Sub	CollapseAllGridItems()
[JScript]	public	function	CollapseAllGridItems();

Description

Collapses all the categories in the **System.Windows.Forms.PropertyGrid** .

The method collapses the categories in the grid. Those categories that have grid items in the category have a plus sign (+) displayed next to that category.

CreatePropertyTab

[C#]	protected	virtual	PropertyTab	CreatePropertyTab(Type	tabType);
[C++]	protected:	virtual	PropertyTab*	CreatePropertyTab(Type*	tabType);
[VB]	Overridable	Protected	Function	CreatePropertyTab(ByVal	tabType As
	Type)		As		PropertyTab
[JScript]	protected	function	CreatePropertyTab(tabType : Type) :	PropertyTab;	

Description

When overridden in a derived class, allows for the creation of a

System.Windows.Forms.Design.PropertyTab .

Return Value: The newly created property tab. Returns **null** in its default implementation.

See **System.Windows.Forms.Design.PropertyTab** for more information on creating property tabs. The type of tab to create.

oooooo)Dispose

[C#] protected override void Dispose(bool disposing);

[C++] protected: void Dispose(bool disposing);

[VB] Overrides Protected Sub Dispose(ByVal disposing As Boolean)

[JScript] protected override function Dispose(disposing : Boolean);

Description

Disposes of the resources (other than memory) used by the **System.Windows.Forms.PropertyGrid** .

Call **System.Windows.Forms.PropertyGrid.Dispose(System.Boolean)** when you are finished using the **System.Windows.Forms.PropertyGrid** . The **System.Windows.Forms.PropertyGrid.Dispose(System.Boolean)** method leaves the **System.Windows.Forms.PropertyGrid** in an unusable state. After calling **System.Windows.Forms.PropertyGrid.Dispose(System.Boolean)** , you must release all references to the **System.Windows.Forms.PropertyGrid** so the memory it was occupying can be reclaimed by garbage collection.

pppppp)ExpandAllGridItems

[C#] public void ExpandAllGridItems();

[C++] public: void ExpandAllGridItems();

[VB] Public Sub ExpandAllGridItems()

[JScript] public function ExpandAllGridItems();

Description

Expands all the categories in the **System.Windows.Forms.PropertyGrid**.

OnComComponentNameChanged

[C#]	protected	void
OnComComponentNameChanged(ComponentRenameEventArgs e);		
[C++]	protected:	void
OnComComponentNameChanged(ComponentRenameEventArgs* e);		
[VB]	Protected Sub	OnComComponentNameChanged(ByVal e As ComponentRenameEventArgs)
[JScript]	protected function	OnComComponentNameChanged(e : ComponentRenameEventArgs);

Description

Raises the ComComponentNameChanged event.

Raising an event invokes the event handler through a delegate. For more information, see . A

System.ComponentModel.Design.ComponentRenameEventArgs that contains the event data.

OnFontChanged

[C#]	protected	override	void	OnFontChanged(EventArgs e);
[C++]	protected:		void	OnFontChanged(EventArgs* e);
[VB]	Overrides	Protected Sub	OnFontChanged(ByVal e As EventArgs)	
[JScript]	protected	override	function	OnFontChanged(e : EventArgs);

Description

sssss) OnGotFocus

```
[C#]    protected    override    void    OnGotFocus(EventArgs    e);
[C++]    protected:    void    OnGotFocus(EventArgs*    e);
[VB]    Overrides Protected Sub OnGotFocus(ByVal e As EventArgs)
[JScript] protected override function OnGotFocus(e : EventArgs);
```

Description

Raises the GotFocus event.

Raising an event invokes the event handler through a delegate. For more information, see .

ttttt) OnHandleCreated

```
[C#]    protected    override    void    OnHandleCreated(EventArgs    e);
[C++]    protected:    void    OnHandleCreated(EventArgs*    e);
[VB]    Overrides Protected Sub OnHandleCreated(ByVal e As EventArgs)
[JScript] protected override function OnHandleCreated(e : EventArgs);
```

Description

Raises the HandleCreated event.

Raising an event invokes the event handler through a delegate. For more information, see .

uuuuuu)OnHandleDestroyed

```
1
2 [C#]    protected    override    void    OnHandleDestroyed(EventArgs    e);
3
4 [C++]    protected:    void    OnHandleDestroyed(EventArgs*    e);
5
6 [VB] Overrides Protected Sub OnHandleDestroyed(ByVal e As EventArgs)
7
8 [JScript] protected override function OnHandleDestroyed(e : EventArgs);
9
```

Description

vvvvvv)OnMouseDown

```
10
11 [C#]    protected    override    void    OnMouseDown(MouseEventArgs    me);
12
13 [C++]    protected:    void    OnMouseDown(MouseEventArgs*    me);
14
15 [VB] Overrides Protected Sub OnMouseDown(ByVal me As MouseEventArgs)
16
17 [JScript] protected override function OnMouseDown(me : MouseEventArgs);
18
```

Description

Raises the **System.Windows.Forms.Control.MouseDown** event.

Raising an event invokes the event handler through a delegate. For more information, see . A **System.Windows.Forms.MouseEventArgs** that contains the event data.

wwwwww)OnMouseMove

```
19
20
21
22
23 [C#]    protected    override    void    OnMouseMove(MouseEventArgs    me);
24
25 [C++]    protected:    void    OnMouseMove(MouseEventArgs*    me);
26
27 [VB] Overrides Protected Sub OnMouseMove(ByVal me As MouseEventArgs)
```

1 [JScript] protected override function OnMouseMove(me : MouseEventArgs);

3 *Description*

4 Raises the **System.Windows.Forms.Control.MouseMove** event.

5 Raising an event invokes the event handler through a delegate. For more
6 information, see . A **System.Windows.Forms.MouseEventArgs** that contains
the event data.

7 *xxxxxx)OnMouseUp*

9 [C#] protected override void OnMouseUp(MouseEventArgs me);

10 [C++] protected: void OnMouseUp(MouseEventArgs* me);

11 [VB] Overrides Protected Sub OnMouseUp(ByVal me As MouseEventArgs)

12 [JScript] protected override function OnMouseUp(me : MouseEventArgs);

14 *Description*

15 Raises the **System.Windows.Forms.Control.MouseUp** event.

16 Raising an event invokes the event handler through a delegate. For more
17 information, see . A **System.Windows.Forms.MouseEventArgs** that contains
the event data.

18 *yyyyyy)OnNotifyPropertyValueUIItemsChanged*

20 [C#] protected void OnNotifyPropertyValueUIItemsChanged(object sender,
21 EventArgs e);

22 [C++] protected: void OnNotifyPropertyValueUIItemsChanged(Object* sender,
23 EventArgs* e);

24 [VB] Protected Sub OnNotifyPropertyValueUIItemsChanged(ByVal sender As
25

```

1 Object,          ByVal          e          As          EventArgs)
2 [JScript] protected function OnNotifyPropertyValueUIItemsChanged(sender :
3 Object,          e          :          EventArgs);
4

```

Description

Raises the NotifyPropertyValueUIItemsChanged event.

Raising an event invokes the event handler through a delegate. For more information, see . The source of the event. A **System.EventArgs** that contains the event data.

zzzzzz) OnPaint

```

11 [C#]      protected      override      void      OnPaint(PaintEventArgs      pevent);
12 [C++]      protected:      void      OnPaint(PaintEventArgs*      pevent);
13 [VB] Overrides Protected Sub OnPaint(ByVal pevent As PaintEventArgs)
14 [JScript] protected override function OnPaint(pevent : PaintEventArgs);
15

```

Description

Raises the Paint event.

Raising an event invokes the event handler through a delegate. For more information, see .

aaaaaaa)OnPropertyTabChanged

```

21 [C#]          protected          virtual          void
22 OnPropertyTabChanged(PropertyTabChangedEventArgs          e);
23 [C++]          protected:          virtual          void
24 OnPropertyTabChanged(PropertyTabChangedEventArgs*          e);
25

```

1 [VB] Overridable Protected Sub OnPropertyTabChanged(ByVal e As
2 PropertyTabChangedEventArgs)

3 [JScript] protected function OnPropertyTabChanged(e :
4 PropertyTabChangedEventArgs);

5
6 *Description*

7 Raises the **System.Windows.Forms.PropertyGrid.PropertyTabChanged**
8 event.

9 Raising an event invokes the event handler through a delegate. For more
10 information, see . A

11 **System.Windows.Forms.PropertyTabChangedEventArgs** that contains the
12 event data.

13 *bbbbbbb)OnPropertyValueChanged*

14 [C#] protected virtual void
15 OnPropertyValueChanged(PropertyValueChangedEventArgs e);

16 [C++] protected: virtual void
17 OnPropertyValueChanged(PropertyValueChangedEventArgs* e);

18 [VB] Overridable Protected Sub OnPropertyValueChanged(ByVal e As
19 PropertyValueChangedEventArgs)

20 [JScript] protected function OnPropertyValueChanged(e :
21 PropertyValueChangedEventArgs);

22 *Description*

23 Raises the **System.Windows.Forms.PropertyGrid.PropertyValueChanged**
24 event.

25 Raising an event invokes the event handler through a delegate. For more
information, see . A

System.Windows.Forms.PropertyValueChangedEventArgs that contains the event data.

ccccccc)OnResize

[C#] protected override void OnResize(EventArgs e);

[C++] protected: void OnResize(EventArgs* e);

[VB] Overrides Protected Sub OnResize(ByVal e As EventArgs)

[JScript] protected override function OnResize(e : EventArgs);

Description

Raises the OnResize event.

Raising an event invokes the event handler through a delegate. For more information, see .

ddddddd)OnSelectedItemChanged

[C#] protected virtual void

OnSelectedItemChanged(SelectedItemChangedEventArgs e);

[C++] protected: virtual void

OnSelectedItemChanged(SelectedItemChangedEventArgs* e);

[VB] Overridable Protected Sub OnSelectedItemChanged(ByVal e As

SelectedItemChangedEventArgs)

[JScript] protected function OnSelectedItemChanged(e :

SelectedItemChangedEventArgs);

Description

1 Raises the
System.Windows.Forms.PropertyGrid.SelectedGridItemChanged event.

2 Raising an event invokes the event handler through a delegate. For more
information, see . A

3 **System.Windows.Forms.SelectedGridItemChangedEventArgs** that
4 contains the event data.

5 *eeeeeee)OnSelectedObjectsChanged*

6
7 [C#] protected virtual void OnSelectedObjectsChanged(EventArgs e);

8 [C++] protected: virtual void OnSelectedObjectsChanged(EventArgs* e);

9 [VB] Overridable Protected Sub OnSelectedObjectsChanged(ByVal e As
10 EventArgs)

11 [JScript] protected function OnSelectedObjectsChanged(e : EventArgs);

12
13 *Description*

14 Raises the
System.Windows.Forms.PropertyGrid.SelectedObjectsChanged event.

15 Raising an event invokes the event handler through a delegate. For more
16 information, see . An **System.EventArgs** that contains the event data.

17 *ffffff) OnSystemColorsChanged*

18
19 [C#] protected override void OnSystemColorsChanged(EventArgs e);

20 [C++] protected: void OnSystemColorsChanged(EventArgs* e);

21 [VB] Overrides Protected Sub OnSystemColorsChanged(ByVal e As EventArgs)

22 [JScript] protected override function OnSystemColorsChanged(e : EventArgs);

23
24 *Description*

ggggggg)OnVisibleChanged

```
1
2 [C#]    protected    override    void    OnVisibleChanged(EventArgs    e);
3
4 [C++]    protected:    void    OnVisibleChanged(EventArgs*    e);
5
6 [VB] Overrides Protected Sub OnVisibleChanged(ByVal e As EventArgs)
7
8 [JScript] protected override function OnVisibleChanged(e : EventArgs);
9
```

Description

Raises the VisibleChanged event.

Raising an event invokes the event handler through a delegate. For more information, see .

hhhhhhh)ProcessDialogKey

```
12
13 [C#]    protected    override    bool    ProcessDialogKey(Keys    keyData);
14
15 [C++]    protected:    bool    ProcessDialogKey(Keys    keyData);
16
17 [VB] Overrides Protected Function ProcessDialogKey(ByVal keyData As Keys)
18 As Boolean
19
20 [JScript] protected override function ProcessDialogKey(keyData : Keys) :
21 Boolean;
22
```

Description

This is a test.

iiiiiii) Refresh

```
23
24 [C#]    public    override    void    Refresh();
25
```

[C++]	public:	void	Refresh();
[VB]	Overrides	Public	Sub Refresh()
[JScript]	public	override	function Refresh();

Description

jjjjjj) RefreshTabs

[C#]	public	void	RefreshTabs(PropertyTabScope	tabScope);
[C++]	public:	void	RefreshTabs(PropertyTabScope	tabScope);
[VB]	Public	Sub	RefreshTabs(ByVal	tabScope As PropertyTabScope)
[JScript]	public	function	RefreshTabs(tabScope	: PropertyTabScope);

Description

Refreshes the property tabs of the specified scope.

The **System.Windows.Forms.PropertyGrid.RefreshTabs(System.ComponentModel.PropertyTabScope)** method first deletes the property tabs of the specified scope, it then requires the objects and documents to rebuild the tabs. Either **System.ComponentModel.PropertyTabScope.Component** or **System.ComponentModel.PropertyTabScope.Document**.

kkkkkkk)ResetSelectedProperty

[C#]	public	void	ResetSelectedProperty();
[C++]	public:	void	ResetSelectedProperty();
[VB]	Public	Sub	ResetSelectedProperty()
[JScript]	public	function	ResetSelectedProperty();

Description

Resets the selected property to its default value.

Typically, the

System.Windows.Forms.PropertyGrid.ResetSelectedProperty method is invoked by right clicking on the property. This method discards changes and attempts to reset the property to its default value.

IIIIII) ScaleCore

[C#] protected override void ScaleCore(float dx, float dy);

[C++] protected: void ScaleCore(float dx, float dy);

[VB] Overrides Protected Sub ScaleCore(ByVal dx As Single, ByVal dy As Single)

[JScript] protected override function ScaleCore(dx : float, dy : float);

Description

mmmmmmm)ShowEventsButton

[C#] protected void ShowEventsButton(bool value);

[C++] protected: void ShowEventsButton(bool value);

[VB] Protected Sub ShowEventsButton(ByVal value As Boolean)

[JScript] protected function ShowEventsButton(value : Boolean);

Description

nnnnnnnn)IComPropertyBrowser.DropDownDone

[C#] void IComPropertyBrowser.DropDownDone();
[C++] void IComPropertyBrowser::DropDownDone();
[VB] Sub DropDownDone() Implements IComPropertyBrowser.DropDownDone
[JScript] function IComPropertyBrowser.DropDownDone();

oooooooo)IComPropertyBrowser.EnsurePendingChangesCommitted

[C#] bool IComPropertyBrowser.EnsurePendingChangesCommitted();
[C++] bool IComPropertyBrowser::EnsurePendingChangesCommitted();
[VB] Function EnsurePendingChangesCommitted() As Boolean Implements
IComPropertyBrowser.EnsurePendingChangesCommitted
[JScript] function IComPropertyBrowser.EnsurePendingChangesCommitted() :
Boolean;

pppppppp)IComPropertyBrowser.HandleF4

[C#] void IComPropertyBrowser.HandleF4();
[C++] void IComPropertyBrowser::HandleF4();
[VB] Sub HandleF4() Implements IComPropertyBrowser.HandleF4
[JScript] function IComPropertyBrowser.HandleF4();

qqqqqqqq)IComPropertyBrowser.LoadState

[C#] void IComPropertyBrowser.LoadState(RegistryKey optRoot);
[C++] void IComPropertyBrowser::LoadState(RegistryKey* optRoot);
[VB] Sub LoadState(ByVal optRoot As RegistryKey) Implements

1 IComPropertyBrowser.LoadState

2 [JScript] function IComPropertyBrowser.LoadState(optRoot : RegistryKey);

3 *rrrrrrr)IComPropertyBrowser.SaveState*

4
5 [C#] void IComPropertyBrowser.SaveState(RegistryKey optRoot);

6 [C++] void IComPropertyBrowser::SaveState(RegistryKey* optRoot);

7 [VB] Sub SaveState(ByVal optRoot As RegistryKey) Implements
8 IComPropertyBrowser.SaveState

9 [JScript] function IComPropertyBrowser.SaveState(optRoot : RegistryKey);

10 *sssssss)UnsafeNativeMethods.IPropertyNotifySink.OnChanged*

11
12 [C#] void UnsafeNativeMethods.IPropertyNotifySink.OnChanged(int dispID);

13 [C++] void UnsafeNativeMethods::IPropertyNotifySink::OnChanged(int dispID);

14 [VB] Sub IPropertyNotifySink.OnChanged(ByVal dispID As Integer) Implements
15 UnsafeNativeMethods.IPropertyNotifySink.OnChanged

16 [JScript] function UnsafeNativeMethods.IPropertyNotifySink.OnChanged(dispID
17 : int);

18 *tttttt) UnsafeNativeMethods.IPropertyNotifySink.OnRequestEdit*

19
20 [C#] int UnsafeNativeMethods.IPropertyNotifySink.OnRequestEdit(int dispID);

21 [C++] int UnsafeNativeMethods::IPropertyNotifySink::OnRequestEdit(int
22 dispID);

23 [VB] Function IPropertyNotifySink.OnRequestEdit(ByVal dispID As Integer) As
24 Integer Implements UnsafeNativeMethods.IPropertyNotifySink.OnRequestEdit

[JScript] function

UnsafeNativeMethods.IPropertyNotifySink.OnRequestEdit(dispid : int) : int;

uuuuuuu)WndProc

[C#] protected override void WndProc(ref Message m);

[C++] protected: void WndProc(Message* m);

[VB] Overrides Protected Sub WndProc(ByRef m As Message)

[JScript] protected override function WndProc(m : Message);

Description

PropertyManager class (System.Windows.Forms)

a) *WndProc*

Description

Maintains a **System.Windows.Forms.Binding** between an object's property and a data-bound control property.

The **System.Windows.Forms.PropertyManager** inherits from the **System.Windows.Forms.BindingManagerBase**, and it is used to maintain the current property of an object, rather than the property of a current object in a list. For this reason, trying to set the **System.Windows.Forms.BindingManagerBase.Position** or **System.Windows.Forms.BindingManagerBase.Count** property for a **System.Windows.Forms.PropertyManager** has no effect. Similarly, the **System.Windows.Forms.BindingManagerBase.AddNew** and **System.Windows.Forms.BindingManagerBase.RemoveAt(System.Int32)** methods are not supported because there is no underlying list of data to add to or delete from.

1 **b) *PropertyManager***

2 *Example Syntax:*

3 **c) *WndProc***

4
5
6 *Description*

7 Initializes a new instance of the **System.Windows.Forms.PropertyManager**
8 class.

9 **d) *Bindings***

10 **e) *Count***

11 **f) *WndProc***

12
13 *Description*

14
15 **g) *Current***

16 **h) *WndProc***

17
18
19 [C#] public override object Current {get;}
20

21 [C++] public: __property virtual Object* get_Current();
22

23 [VB] Overrides Public ReadOnly Property Current As Object
24

25 [JScript] public function get Current() : Object;

26 *Description*

27 Gets the object to which the data-bound property belongs.

The **System.Windows.Forms.PropertyManager.Current** property returns the data source of a data-bound relationship.

i) *Position*

j) *WndProc*

[C#] public override int Position {get; set;}

[C++] public: __property virtual int get_Position();public: __property virtual void set_Position(int);

[VB] Overrides Public Property Position As Integer

[JScript] public function get Position() : int;public function set Position(int);

Description

k) *AddNew*

[C#] public override void AddNew();

[C++] public: void AddNew();

[VB] Overrides Public Sub AddNew()

[JScript] public override function AddNew();

Description

Adds a new row.

l) *CancelCurrentEdit*

[C#] public override void CancelCurrentEdit();

[C++]	public:	void	CancelCurrentEdit();	
[VB]	Overrides	Public	Sub	CancelCurrentEdit()
[JScript]	public	override	function	CancelCurrentEdit();

Description

Cancels the current **System.Windows.Forms.Binding** between a data binding and a data-bound property.

m) *EndCurrentEdit*

[C#]	public	override	void	EndCurrentEdit();
[C++]	public:	void	EndCurrentEdit();	
[VB]	Overrides	Public	Sub	EndCurrentEdit()
[JScript]	public	override	function	EndCurrentEdit();

Description

Ends the current **System.Windows.Forms.Binding** between a data binding and a data-bound property.

n) *GetItemProperties*

[C#]	public	override	PropertyDescriptorCollection	GetItemProperties();	
[C++]	public:	PropertyDescriptorCollection*	GetItemProperties();		
[VB]	Overrides	Public	Function	GetItemProperties()	As
			PropertyDescriptorCollection		
[JScript]	public	override	function	GetItemProperties()	:
			PropertyDescriptorCollection;		

1
2 *Description*

3 Gets the property descriptors for the managed object.

4 *Return Value:* A PropertyDescriptorCollection containing the PropertyDescriptor objects for the managed object.

5 o) *GetListName*

6
7 [C#] protected internal override string GetListName(ArrayList listAccessors);

8 [C++] protected public: String* GetListName(ArrayList* listAccessors);

9 [VB] Overrides Protected Friend Dim Function GetListName(ByVal listAccessors

10 As ArrayList) As String

11 [JScript] package override function GetListName(listAccessors : ArrayList) :

12 String;

13
14 *Description*

15 Gets the name of the underlying data list.

16 p) *OnCurrentChanged*

17
18 [C#] protected internal override void OnCurrentChanged(EventArgs ea);

19 [C++] protected public: void OnCurrentChanged(EventArgs* ea);

20 [VB] Overrides Protected Friend Dim Sub OnCurrentChanged(ByVal ea As

21 EventArgs)

22 [JScript] package override function OnCurrentChanged(ea : EventArgs);

23
24 *Description*

25

Raises the
System.Windows.Forms.BindingManagerBase.CurrentChanged event.

q) RemoveAt

[C#] public override void RemoveAt(int index);
[C++] public: void RemoveAt(int index);
[VB] Overrides Public Sub RemoveAt(ByVal index As Integer)
[JScript] public override function RemoveAt(index : int);

Description

Deletes a specified row.

r) ResumeBinding

[C#] public override void ResumeBinding();
[C++] public: void ResumeBinding();
[VB] Overrides Public Sub ResumeBinding()
[JScript] public override function ResumeBinding();

Description

Resumes the data binding between an data source and a data-bound property.

s) SuspendBinding

[C#] public override void SuspendBinding();
[C++] public: void SuspendBinding();
[VB] Overrides Public Sub SuspendBinding()

[JScript] public override function SuspendBinding();

Description

Suspends the data binding between an data source and a data-bound property.

System.Windows.Forms.PropertyManager.SuspendBinding and **System.Windows.Forms.PropertyManager.ResumeBinding** are two methods that allow the temporary suspension and resumption of data binding. You typically suspend data binding if the user must be allowed to make several edits to data fields before validation occurs. For example, if one field must be changed in accordance with a second, but where validating the first field would cause the second field to be in error.

t) UpdateIsBinding

[C#]	protected	override	void	UpdateIsBinding();
[C++]	protected:		void	UpdateIsBinding();
[VB]	Overrides	Protected	Sub	UpdateIsBinding()
[JScript]	protected	override	function	UpdateIsBinding();

Description

Updates the ccurrent **System.Windows.Forms.Binding** betwwen a data binding and a data-bound property.

PropertySort enumeration (System.Windows.Forms)

a) UpdateIsBinding

Description

Specifies how properties are sorted in the **System.Windows.Forms.PropertyGrid** .

Use the members of this enumeration to set the value of the **System.Windows.Forms.PropertyGrid.PropertySort** property of the **System.Windows.Forms.PropertyGrid**.

b) UpdateIsBinding

[C#]	public	const	PropertySort	Alphabetical;
[C++]	public:	const	PropertySort	Alphabetical;
[VB]	Public	Const	Alphabetical	As PropertySort
[JScript]	public	var	Alphabetical	: PropertySort;

Description

Properties are sorted in an alphabetical list.

c) UpdateIsBinding

[C#]	public	const	PropertySort	Categorized;
[C++]	public:	const	PropertySort	Categorized;
[VB]	Public	Const	Categorized	As PropertySort
[JScript]	public	var	Categorized	: PropertySort;

Description

Properties are displayed according to their category in a group. The categories are defined by the properties themselves.

d) UpdateIsBinding

[C#]	public	const	PropertySort	CategorizedAlphabetical;
[C++]	public:	const	PropertySort	CategorizedAlphabetical;

[VB]	Public	Const	CategorizedAlphabetical	As	PropertySort
[JScript]	public	var	CategorizedAlphabetical	:	PropertySort;

Description

Properties are displayed according to their category in a group. The properties are further sorted alphabetically within the group. The categories are defined by the properties themselves.

e) UpdateIsBinding

[C#]	public	const	PropertySort	NoSort;
[C++]	public:	const	PropertySort	NoSort;
[VB]	Public	Const	NoSort	As PropertySort
[JScript]	public	var	NoSort	: PropertySort;

Description

Properties are displayed in the order in which they are retrieved from the **System.ComponentModel.TypeDescriptor** .

PropertyTabChangedEventArgs class (System.Windows.Forms)

a) ToString

Description

Provides data for the **System.Windows.Forms.PropertyGrid.PropertyTabChanged** event of a **System.Windows.Forms.PropertyGrid** .

The **System.Windows.Forms.PropertyGrid.PropertyTabChanged** event occurs when the user selects a new property tab in the **System.Windows.Forms.PropertyGrid** .

b) *PropertyTabChangedEventArgs*

Example Syntax:

c) *ToString*

```
[C#] public PropertyTabChangedEventArgs(PropertyTab oldTab, PropertyTab
newTab);
```

```
[C++] public: PropertyTabChangedEventArgs(PropertyTab* oldTab,
PropertyTab* newTab);
```

```
[VB] Public Sub New(ByVal oldTab As PropertyTab, ByVal newTab As
PropertyTab)
```

```
[JScript] public function PropertyTabChangedEventArgs(oldTab : PropertyTab,
newTab : PropertyTab);
```

Description

Initializes a new instance of the **System.Windows.Forms.PropertyTabChangedEventArgs** class. The
Previously selected property tab. The newly selected property tab.

d) *NewTab*

e) *ToString*

```
[C#] public PropertyTab NewTab {get;}
```

```
[C++] public: __property PropertyTab* get_NewTab();
```

```
[VB] Public ReadOnly Property NewTab As PropertyTab
```

```
[JScript] public function get NewTab() : PropertyTab;
```

Description

Gets the new **System.Windows.Forms.Design.PropertyTab** selected.

f) OldTab

g) ToString

[C#] public PropertyTab OldTab {get;}

[C++] public: __property PropertyTab* get_OldTab();

[VB] Public ReadOnly Property OldTab As PropertyTab

[JScript] public function get OldTab() : PropertyTab;

Description

Gets the old **System.Windows.Forms.Design.PropertyTab** selected.

PropertyTabChangedEventHandler delegate (System.Windows.Forms)

a) ToString

Description

Represents the method that will handle the **System.Windows.Forms.PropertyGrid.PropertyTabChanged** event of a **System.Windows.Forms.PropertyGrid**. The source of the event. A **System.Windows.Forms.PropertyTabChangedEventArgs** that contains the event data.

When you create a **System.Windows.Forms.PropertyTabChangedEventHandler** delegate, you identify the method that will handle the event. To associate the event with your event handler, add an instance of the delegate to the event. The event handler is called whenever the event occurs, unless you remove the delegate.

PropertyGrid.PropertyTabCollection class (System.Windows.Forms)

a) *ToString*

Description

Contains a collection of **System.Windows.Forms.Design.PropertyTab** objects.

b) *Count*

c) *ToString*

```
[C#]          public          int          Count          {get;}
[C++]          public:          __property          int          get_Count();
[VB]    Public    ReadOnly    Property    Count    As    Integer
[JScript]    public    function    get    Count()    :    int;
```

Description

Gets the number of property tabs in the collection.

d) *Item*

e) *ToString*

```
[C#]          public          PropertyTab          this[int          index]          {get;}
[C++]          public:          __property          PropertyTab*          get_Item(int          index);
[VB]    Public    Default    ReadOnly    Property    Item(ByVal index As Integer) As
PropertyTab
[JScript]    returnValue          =          PropertyTabCollectionObject.Item(index);
```

Description

Gets the **System.Windows.Forms.Design.PropertyTab** at the specified index. The index of the **System.Windows.Forms.Design.PropertyTab** to return.

f) AddTabType

```
[C#]      public      void      AddTabType(Type      propertyTabType);
[C++]     public:     void      AddTabType(Type*      propertyTabType);
[VB]      Public      Sub      AddTabType(ByVal      propertyTabType      As      Type)
[JavaScript] public function AddTabType(propertyTabType : Type); Adds a property
tab                to                the                collection.
```

Description

Adds a property tab of the specified type to the collection.

The property tab is added to the collection with a scope of **System.ComponentModel.PropertyTabScope.Global**. The property tab type to add to the grid.

g) AddTabType

```
[C#]      public      void      AddTabType(Type      propertyTabType,      PropertyTabScope
tabScope);
[C++]     public:     void      AddTabType(Type*      propertyTabType,      PropertyTabScope
tabScope);
[VB]      Public      Sub      AddTabType(ByVal      propertyTabType      As      Type,      ByVal      tabScope
As                PropertyTabScope)
```

1 [JScript] public function AddTabType(propertyTabType : Type, tabScope :
2 PropertyTabScope);

3
4 *Description*

5 Adds a property tab of the specified type and with the specified scope to the
6 collection. The property tab type to add to the grid. One of the
System.ComponentModel.PropertyTabScope values.

7 *h) Clear*

8
9 [C#] public void Clear(PropertyTabScope tabScope);

10 [C++] public: void Clear(PropertyTabScope tabScope);

11 [VB] Public Sub Clear(ByVal tabScope As PropertyTabScope)

12 [JScript] public function Clear(tabScope : PropertyTabScope);

13
14 *Description*

15 Removes all the property tabs of the specified scope from the collection.

16 This method clears the tabs of the specified scope or smaller. The *tabScope*
parameter must be

17 **System.ComponentModel.PropertyTabScope.Component** or
18 **System.ComponentModel.PropertyTabScope.Document** . The scope of
the tabs to clear.

19 *i) GetEnumerator*

20
21 [C#] public IEnumerator GetEnumerator();

22 [C++] public: __sealed IEnumerator* GetEnumerator();

23 [VB] NotOverridable Public Function GetEnumerator() As IEnumerator

24 [JScript] public function GetEnumerator() : IEnumerator;

Description

Returns an enumeration of all the property tabs in the collection.

Return Value: An **System.Collections.IEnumerator** for the **System.Windows.Forms.PropertyGrid.PropertyTabCollection** .

This method creates an enumerator that contains a snapshot of the collection. You can change the collection by changing the enumerator; however, multiple enumerators can simultaneously access the same collection. Changing the collection (either directly or through another enumerator) can cause **System.Collections.IEnumerator.Current** or **System.Collections.IEnumerator.MoveNext** to throw an exception.

j) RemoveTabType

[C#] public void RemoveTabType(Type propertyTabType);

[C++] public: void RemoveTabType(Type* propertyTabType);

[VB] Public Sub RemoveTabType(ByVal propertyTabType As Type)

[JScript] public function RemoveTabType(propertyTabType : Type);

Description

Removes the specified tab type from the collection.

Removing the tab from the collection removes it from the property grid. The tab type to remove from the collection.

k) ICollection.CopyTo

[C#] void ICollection.CopyTo(Array dest, int index);

[C++] void ICollection::CopyTo(Array* dest, int index);

[VB] Sub CopyTo(ByVal dest As Array, ByVal index As Integer) Implements

1 ICollection.CopyTo

2 [JScript] function ICollection.CopyTo(dest : Array, index : int);

3 PropertyValueChangedEventArgs class (System.Windows.Forms)

4 a) *ToString*

7 *Description*

8 Provides data for the
9 **System.Windows.Forms.PropertyGrid.PropertyValueChanged** event of a
10 **System.Windows.Forms.PropertyGrid** .

11 The **System.Windows.Forms.PropertyGrid.PropertyValueChanged** event
12 occurs when the user changes the value of a property, which is specified as a
13 **System.Windows.Forms.GridItem** , in a
14 **System.Windows.Forms.PropertyGrid** .

15 b) *PropertyValueChangedEventArgs*

16 *Example Syntax:*

17 c) *ToString*

18 [C#] public PropertyValueChangedEventArgs(GridItem changedItem, object
19 oldValue);

20 [C++] public: PropertyValueChangedEventArgs(GridItem* changedItem, Object*
21 oldValue);

22 [VB] Public Sub New(ByVal changedItem As GridItem, ByVal oldValue As
23 Object)

24 [JScript] public function PropertyValueChangedEventArgs(changedItem :
25 GridItem, oldValue : Object);

Description

Initializes a new instance of the **System.Windows.Forms.PropertyValueChangedEventArgs** class. The item in the grid that changed. The old property value.

d) *ChangedItem*

e) *ToString*

```
[C#]          public          GridItem          ChangedItem          {get;}
[C++]          public:          __property          GridItem*          get_ChangedItem();
[VB]    Public    ReadOnly    Property    ChangedItem    As    GridItem
[JScript]    public    function    get    ChangedItem()    :    GridItem;
```

Description

Gets the **System.Windows.Forms.GridItem** that was changed.

f) *OldValue*

g) *ToString*

```
[C#]          public          object          OldValue          {get;}
[C++]          public:          __property          Object*          get_OldValue();
[VB]    Public    ReadOnly    Property    OldValue    As    Object
[JScript]    public    function    get    OldValue()    :    Object;
```

Description

The value of the grid item before it was changed.

This property provides you with the value of the property as it was before the change was applied. You can find the new value by querying the property grid with the specified **System.Windows.Forms.PropertyValueChangedEventArgs.ChangedItem**.

PropertyValueChangedEventHandler delegate (System.Windows.Forms)

a) *ToString*

Description

The event handler class that is invoked when a property in the grid is modified by the user.

QueryAccessibilityHelpEventArgs class (System.Windows.Forms)

a) *ToString*

Description

Provides data for the **System.Windows.Forms.Control.QueryAccessibilityHelp** event.

b) *QueryAccessibilityHelpEventArgs*

Example Syntax:

c) *ToString*

[C#] public QueryAccessibilityHelpEventArgs();

[C++] public: QueryAccessibilityHelpEventArgs();

[VB] Public Sub New()

[JScript] public function QueryAccessibilityHelpEventArgs(); Initializes a new

instance of the **System.Windows.Forms.QueryAccessibilityHelpEventArgs** class.

Description

Initializes a new instance of the **System.Windows.Forms.QueryAccessibilityHelpEventArgs** class.

d) QueryAccessibilityHelpEventArgs

Example Syntax:

e) ToString

[C#] public QueryAccessibilityHelpEventArgs(string helpNamespace, string helpString, string helpKeyword);

[C++] public: QueryAccessibilityHelpEventArgs(String* helpNamespace, String* helpString, String* helpKeyword);

[VB] Public Sub New(ByVal helpNamespace As String, ByVal helpString As String, ByVal helpKeyword As String)

[JScript] public function QueryAccessibilityHelpEventArgs(helpNamespace : String, helpString : String, helpKeyword : String);

Description

Initializes a new instance of the **System.Windows.Forms.QueryAccessibilityHelpEventArgs** class. The file containing Help for the **System.Windows.Forms.AccessibleObject** . The string defining what Help to get for the **System.Windows.Forms.AccessibleObject** . The keyword to associate with Help request for the **System.Windows.Forms.AccessibleObject** .

1 *f) HelpKeyword*

2 *g) ToString*

3
4 [C#] public string HelpKeyword {get; set;}

5 [C++] public: __property String* get_HelpKeyword();public: __property void
6 set_HelpKeyword(String*);

7 [VB] Public Property HelpKeyword As String

8 [JScript] public function get HelpKeyword() : String;public function set
9 HelpKeyword(String);

10
11 *Description*

12 Gets or sets the help keyword for the specified control.

13 *h) HelpNamespace*

14 *i) ToString*

15
16 [C#] public string HelpNamespace {get; set;}

17 [C++] public: __property String* get_HelpNamespace();public: __property void
18 set_HelpNamespace(String*);

19 [VB] Public Property HelpNamespace As String

20 [JScript] public function get HelpNamespace() : String;public function set
21 HelpNamespace(String);

22
23 *Description*

24 Gets or sets a value specifying the name of the help file.

j) *HelpString*

k) *ToString*

[C#] public string HelpString {get; set;}

[C++] public: __property String* get_HelpString();public: __property void
set_HelpString(String*);

[VB] Public Property HelpString As String

[JScript] public function get HelpString() : String;public function set
HelpString(String);

Description

Gets or sets the string defining what help to get for the
System.Windows.Forms.AccessibleObject .

QueryAccessibilityHelpEventHandler delegate (System.Windows.Forms)

a) *ToString*

Description

Represents the method that will handle the QueryAccessibilityHelp event of an
System.Windows.Forms.AccessibilityObject .

When you create a
System.Windows.Forms.QueryAccessibilityHelpEventHandler delegate,
you identify the method that will handle the event. To associate the event with
your event handler, add an instance of the delegate to the event. The event
handler is called whenever the event occurs, unless you remove the delegate.
For more information about event handler delegates, see .

QueryContinueDragEventArgs class (System.Windows.Forms)

a) *ToString*

Description

Provides data for the **System.Windows.Forms.Control.QueryContinueDrag** event.

The **System.Windows.Forms.Control.QueryContinueDrag** event occurs during a drag-and-drop operation and allows the drag source to determine whether the drag-and-drop operation should be canceled. A

System.Windows.Forms.QueryContinueDragEventArgs object specifies whether and how the drag-and-drop operation should proceed, whether any modifier keys are pressed, and whether the user has pressed the ESC key.

b) *QueryContinueDragEventArgs*

Example Syntax:

c) *ToString*

```
[C#] public QueryContinueDragEventArgs(int keyState, bool escapePressed,  
DragAction action);
```

```
[C++] public: QueryContinueDragEventArgs(int keyState, bool escapePressed,  
DragAction action);
```

```
[VB] Public Sub New(ByVal keyState As Integer, ByVal escapePressed As  
Boolean, ByVal action As DragAction)
```

```
[JScript] public function QueryContinueDragEventArgs(keyState : int,  
escapePressed : Boolean, action : DragAction);
```

Description

Initializes a new instance of the

System.Windows.Forms.QueryContinueDragEventArgs class. The current state of the SHIFT, CTRL, and ALT keys. **true** if the ESC key was pressed; otherwise, **false**. One of the **System.Windows.Forms.DragAction** values.

d) *Action*

e) *ToString*

```
[C#]      public      DragAction      Action      {get;      set;}
```

```
[C++] public: __property DragAction get_Action();public: __property void  
set_Action(DragAction);
```

```
[VB]      Public      Property      Action      As      DragAction
```

```
[JScript] public function get Action() : DragAction;public function set  
Action(DragAction);
```

Description

Gets or sets the status of a drag-and-drop operation.

f) *EscapePressed*

g) *ToString*

```
[C#]      public      bool      EscapePressed      {get;}
```

```
[C++] public:      __property      bool      get_EscapePressed();
```

```
[VB]      Public      ReadOnly      Property      EscapePressed      As      Boolean
```

```
[JScript] public function get EscapePressed() : Boolean;
```

Description

Gets whether the user pressed the ESC key.

h) *KeyState*

i) *ToString*

[C#]	public	int	KeyState	{get;}
[C++]	public:	__property	int	get_KeyState();
[VB]	Public	ReadOnly	Property	KeyState As Integer
[JScript]	public	function	get	KeyState() : int;

Description

Gets the current state of the SHIFT, CTRL, and ALT keys.

QueryContinueDragEventHandler delegate (System.Windows.Forms)

a) *ToString*

Description

Represents the method that will handle the **System.Windows.Forms.Control.QueryContinueDrag** event of a **System.Windows.Forms.Control** . The source of an event. A **System.Windows.Forms.QueryContinueDragEventArgs** that contains the event data.

When you create a **System.Windows.Forms.QueryContinueDragEventHandler** delegate, you identify the method that will handle the event. To associate the event with your event handler, add an instance of the delegate to the event. The event handler is called whenever the event occurs, unless you remove the delegate. For more information about handling events with delegates, see .

RadioButton class (System.Windows.Forms)

a) *ToString*

Description

Represents a Windows radio button.

The **System.Windows.Forms.RadioButton** control may display text, an **System.Drawing.Image** , or both.

b) *RadioButton*

Example Syntax:

c) *ToString*

[C#]	public	RadioButton();
[C++]	public:	RadioButton();
[VB]	Public	Sub New()
[JScript]	public	function RadioButton();

Description

Initializes a new instance of the **System.Windows.Forms.RadioButton** class.

The default view of the **System.Windows.Forms.RadioButton** has its text aligned to the right of the button and the **System.Windows.Forms.RadioButton.AutoCheck** property is set to **true** .

- d) *AccessibilityObject*
- e) *AccessibleDefaultActionDescription*
- f) *AccessibleDescription*
- g) *AccessibleName*
- h) *AccessibleRole*
- i) *AllowDrop*
- j) *Anchor*
- k) *Appearance*
- l) *ToString*

Description

Gets or set the value that determines the appearance of the radio button control.

If the **System.Windows.Forms.RadioButton.Appearance** value is set to **Normal** , then the **System.Windows.Forms.RadioButton** control is drawn with a circular check box. If the value is set to **Appearance.Button** , then the **System.Windows.Forms.RadioButton** control is drawn as a button that may be toggled to an up or down state. Either type may display text, an image, or both.

m) *AutoCheck*

n) *ToString*

[C#] public bool AutoCheck {get; set;}

[C++] public: __property bool get_AutoCheck();public: __property void
set_AutoCheck(bool);

[VB] Public Property AutoCheck As Boolean

[JScript] public function get AutoCheck() : Boolean; public function set AutoCheck(Boolean);

Description

Gets or sets a value indicating whether the **System.Windows.Forms.RadioButton.Checked** value and the appearance of the control automatically change when the control is clicked.

If the **System.Windows.Forms.RadioButton.Checked** value is set to **false**, the radio button portion of the control must be checked in code in the **System.Windows.Forms.Control.Click** event handler. In addition, if the **System.Windows.Forms.RadioButton** is part of a **System.Windows.Forms.RadioButton** control group, this property ensures that only one of the controls is checked at a given time.

o) BackColor

p) BackgroundImage

q) BindingContext

r) Bottom

s) Bounds

t) CanFocus

u) CanSelect

v) Capture

w) CausesValidation

x) CheckAlign

y) ToString

Description

Gets or sets the location of the check box portion of the radio button control.

z) *Checked*

aa) *ToString*

[C#] public bool Checked {get; set;}

[C++] public: __property bool get_Checked();public: __property void
set_Checked(bool);

[VB] Public Property Checked As Boolean

[JScript] public function get Checked() : Boolean;public function set
Checked(Boolean);

Description

Gets or sets a value indicating whether the control is checked.

1 **bb) ClientRectangle**

2 **cc) ClientSize**

3 **dd) CompanyName**

4 **ee) Container**

5 **ff) ContainsFocus**

6 **gg) ContextMenu**

7 **hh) Controls**

8 **ii) Created**

9 **jj) CreateParams**

10 **kk) ToString**

11
12
13 *Description*

14 **ll) Cursor**

15 **mm) DataBindings**

16 **nn) DefaultImeMode**

17 **oo) DefaultSize**

18 **pp) ToString**

19
20
21
22 *Description*

23 Deriving classes can override this to configure a default size for their control.
24 This is more efficient than setting the size in the control's constructor.
25

qq) DesignMode

rr) DisplayRectangle

ss) Disposing

tt) Dock

uu) Enabled

vv) Events

ww) FlatStyle

xx) Focused

yy) Font

zz) FontHeight

aaa) ForeColor

bbb) Handle

ccc) HasChildren

ddd) Height

eee) Image

fff) ImageAlign

ggg) ImageIndex

hhh) ImageList

iii) ImeMode

jjj) InvokeRequired

kkk) IsAccessible

lll) IsDefault

mmm) IsDisposed

MS1-861US.APP
509-324-9256
lee@hayes pfc

1	<i>nnn) IsHandleCreated</i>
2	<i>ooo) Left</i>
3	<i>ppp) Location</i>
4	<i>qqq) Name</i>
5	<i>rrr) Parent</i>
6	<i>sss) ProductName</i>
7	<i>ttt) ProductVersion</i>
8	<i>uuu) RecreatingHandle</i>
9	<i>vvv) Region</i>
10	<i>www) RenderRightToLeft</i>
11	<i>xxx) ResizeRedraw</i>
12	<i>yyy) Right</i>
13	<i>zzz) RightToLeft</i>
14	<i>aaaa) ShowFocusCues</i>
15	<i>bbbb) ShowKeyboardCues</i>
16	<i>cccc) Site</i>
17	<i>dddd) Size</i>
18	<i>eeee) TabIndex</i>
19	<i>ffff) TabStop</i>
20	<i>gggg) ToString</i>
21	
22	
23	
24	<i>Description</i>
25	

Gets or sets a value indicating whether the user can give the focus to this control using the TAB key.

hhh) Tag

iii) Text

jjj) TextAlign

kkk) ToString

Description

Gets or sets the alignment of the text on the radiobutton control.

lll) Top

mmm)TopLevelControl

nnn) Visible

ooo) Width

pppp) WindowTarget

qqqq) ToString

Description

Occurs when the **System.Windows.Forms.RadioButton.Appearance** property value changes.

For more information about handling events, see .

rrrr) ToString

Description

Occurs when the value of the **System.Windows.Forms.RadioButton.Checked** property changes.

For more information about handling events, see .

ssss) CreateAccessibilityInstance

[C#] protected override AccessibleObject CreateAccessibilityInstance();

[C++] protected: AccessibleObject* CreateAccessibilityInstance();

[VB] Overrides Protected Function CreateAccessibilityInstance() As AccessibleObject

[JScript] protected override function CreateAccessibilityInstance() : AccessibleObject;

Description

Constructs the new instance of the accessibility object for this control. Subclasses should not call base.CreateAccessibilityObject.

tttt) OnCheckedChanged

[C#] protected virtual void OnCheckedChanged(EventArgs e);

[C++] protected: virtual void OnCheckedChanged(EventArgs* e);

[VB] Overridable Protected Sub OnCheckedChanged(ByVal e As EventArgs)

[JScript] protected function OnCheckedChanged(e : EventArgs);

Description

Raises the **System.Windows.Forms.CheckBox.CheckedChanged** event.

Raising an event invokes the event handler through a delegate. For more information, see . An **System.EventArgs** that contains the event data.

uuuu) OnClick

```
[C#]      protected      override      void      OnClick(EventArgs      e);
```

```
[C++]      protected:      void      OnClick(EventArgs*      e);
```

```
[VB] Overrides Protected Sub OnClick(ByVal e As EventArgs)
```

```
[JScript] protected override function OnClick(e : EventArgs);
```

Description

We override this to implement the autoCheck functionality.

vvvv) OnEnter

```
[C#]      protected      override      void      OnEnter(EventArgs      e);
```

```
[C++]      protected:      void      OnEnter(EventArgs*      e);
```

```
[VB] Overrides Protected Sub OnEnter(ByVal e As EventArgs)
```

```
[JScript] protected override function OnEnter(e : EventArgs);
```

Description

www)OnHandleCreated

```
[C#]      protected      override      void      OnHandleCreated(EventArgs      e);
```

1 [C++] protected: void OnHandleCreated(EventArgs* e);

2 [VB] Overrides Protected Sub OnHandleCreated(ByVal e As EventArgs)

3 [JScript] protected override function OnHandleCreated(e : EventArgs);

5 *Description*

6 *xxxx) OnMouseUp*

8 [C#] protected override void OnMouseUp(MouseEventArgs mevent);

9 [C++] protected: void OnMouseUp(MouseEventArgs* mevent);

10 [VB] Overrides Protected Sub OnMouseUp(ByVal mevent As MouseEventArgs)

11 [JScript] protected override function OnMouseUp(mevent : MouseEventArgs);

13 *Description*

14 Raises the
15 **System.Windows.Forms.ButtonBase.OnMouseUp(System.Windows.For
ms.MouseEventArgs)** event.

16 Raising an event invokes the event handler through a delegate. For an overview,
17 see XXX. A **System.Windows.Forms.MouseEventArgs** that contains the
event data.

18 *yyyy) PerformClick*

20 [C#] public void PerformClick();

21 [C++] public: void PerformClick();

22 [VB] Public Sub PerformClick()

23 [JScript] public function PerformClick();

Description

Generates a **System.Windows.Forms.Control.Click** event for the button, simulating a click by a user.

zzzz) ProcessMnemonic

```
[C#]    protected    override    bool    ProcessMnemonic(char    charCode);
[C++]    protected:    bool    ProcessMnemonic(__wchar_t    charCode);
[VB] Overrides Protected Function ProcessMnemonic(ByVal charCode As Char)
As
Boolean
[JavaScript] protected override function ProcessMnemonic(charCode : Char) :
Boolean;
```

Description

aaaaa) ToString

```
[C#]          public          override          string          ToString();
[C++]          public:          String*          ToString();
[VB] Overrides Public Function ToString() As String
[JavaScript] public override function ToString() : String;
```

Description

Returns a string representation for this control.

Return Value: String Returns a string representation for this control.

RadioButton.RadioButtonAccessibleObject class (System.Windows.Forms)

a) *WndProc*

Description

b) *RadioButton.RadioButtonAccessibleObject*

Example Syntax:

c) *WndProc*

[C#] public RadioButton.RadioButtonAccessibleObject(RadioButton owner);

[C++] public: RadioButtonAccessibleObject(RadioButton* owner);

[VB] Public Sub New(ByVal owner As RadioButton)

[JScript] public function RadioButton.RadioButtonAccessibleObject(owner :
RadioButton);

Description

d) *Bounds*

e) *DefaultAction*

f) *WndProc*

Description

- g) Description*
- h) Handle*
- i) Help*
- j) KeyboardShortcut*
- k) Name*
- l) Owner*
- m) Parent*
- n) Role*
- o) WndProc*

Description

- p) State*
- q) WndProc*

```
[C#]      public      override      AccessibleStates      State      {get;}
[C++]     public:     __property     virtual      AccessibleStates      get_State();
[VB]      Overrides   Public   ReadOnly   Property   State   As   AccessibleStates
[JScript] public      function      get      State()      :      AccessibleStates;
```

Description

1 *r) Value*

2 *s) DoDefaultAction*

3
4 [C#] public override void DoDefaultAction();

5 [C++] public: void DoDefaultAction();

6 [VB] Overrides Public Sub DoDefaultAction()

7 [JScript] public override function DoDefaultAction();

8
9 *Description*

10 RichTextBox class (System.Windows.Forms)

11
12 *a) UseStdAccessibleObjects*

13
14
15 *Description*

16 Represents a Windows rich text box control.

17 The **System.Windows.Forms.RichTextBox** control allows the user to enter
18 and edit text while also providing more advanced formatting features than the
19 standard **System.Windows.Forms.TextBox** control. Text can be assigned
20 directly to the control, or can be loaded from a Rich Text Format (RTF) or plain
21 text file. The text within the control can be assigned character and paragraph
22 formatting.

23 *b) RichTextBox*

24 *Example Syntax:*

25 *c) UseStdAccessibleObjects*

26 [C#] public RichTextBox();

1	[C++]	public:	RichTextBox();
2	[VB]	Public	Sub New()
3	[JScript]	public	function RichTextBox();

Description

Initializes a new instance of the **System.Windows.Forms.RichTextBox** class.

By default, the **System.Windows.Forms.TextBoxBase.Multiline** property of the control is set to **true** .

- d) *AcceptsTab*
- e) *AccessibilityObject*
- f) *AccessibleDefaultActionDescription*
- g) *AccessibleDescription*
- h) *AccessibleName*
- i) *AccessibleRole*
- j) *AllowDrop*
- k) *UseStdAccessibleObjects*

Description

Gets or sets a value indicating whether the control will allow drag and drop operations.

1) *Anchor*

m) *AutoSize*

n) *UseStdAccessibleObjects*

Description

Gets or sets a value indicating whether the size of the **System.Windows.Forms.RichTextBox** automatically adjusts when the font assigned to the control is changed.

When this property is set to **true** , the **System.Windows.Forms.RichTextBox** resizes its height based on the size of the font assigned to the control. You can use this property to ensure that the user can read text assigned to the control regardless of the font.

o) *AutoWordSelection*

p) *UseStdAccessibleObjects*

[C#] public bool AutoWordSelection {get; set;}

[C++] public: __property bool get_AutoWordSelection();public: __property void set_AutoWordSelection(bool);

[VB] Public Property AutoWordSelection As Boolean

[JScript] public function get AutoWordSelection() : Boolean;public function set AutoWordSelection(Boolean);

Description

Gets or sets a value indicating whether automatic word selection is enabled.

If this property is set to **true** , selecting any part of the text in the control results in the selection of the entire word. For example, if

System.Windows.Forms.RichTextBox.AutoWordSelection is set to **true** ,

the user can double-click on any part of a word in the control and the entire word is selected automatically.

q) *BackColor*

r) *BackgroundImage*

s) *UseStdAccessibleObjects*

Description

t) *BindingContext*

u) *BorderStyle*

v) *Bottom*

w) *Bounds*

x) *BulletIndent*

y) *UseStdAccessibleObjects*

Description

Gets or sets the indentation used in the **System.Windows.Forms.RichTextBox** control when the bullet style is applied to the text.

To apply the bullet style to a paragraph of text, set the **System.Windows.Forms.RichTextBox.SelectionBullet** property to **true** and then set the **System.Windows.Forms.RichTextBox.BulletIndent** property to the number of pixels that the text should be indented. The paragraph will have the bullet style applied to it with the specified amount of indentation after the bullet. This property only affects the current paragraph within the control's text and the currently selected bullet in a list of bulleted items. To apply a different indentation level to an entire list of bulleted items, all of the text of

the bulleted items must be selected before setting the
System.Windows.Forms.RichTextBox.BulletIndent property.

z) *CanFocus*

aa) *CanRedo*

bb) *UseStdAccessibleObjects*

Description

Gets a value indicating whether there are actions that have occurred within the
System.Windows.Forms.RichTextBox that can be reapplied.

You can use this property to determine whether the last operation undone in the
System.Windows.Forms.RichTextBox can be reapplied using the
System.Windows.Forms.RichTextBox.Redo method.

1	<i>cc) CanSelect</i>
2	<i>dd) CanUndo</i>
3	<i>ee) Capture</i>
4	<i>ff) CausesValidation</i>
5	<i>gg) ClientRectangle</i>
6	<i>hh) ClientSize</i>
7	<i>ii) CompanyName</i>
8	<i>jj) Container</i>
9	<i>kk) ContainsFocus</i>
10	<i>ll) ContextMenu</i>
11	<i>mm) Controls</i>
12	<i>nn) Created</i>
13	<i>oo) CreateParams</i>
14	<i>pp) UseStdAccessibleObjects</i>
15	<i>qq) Cursor</i>
16	<i>rr) DataBindings</i>
17	<i>ss) DefaultImeMode</i>
18	<i>tt) DefaultSize</i>
19	<i>uu) UseStdAccessibleObjects</i>
20	
21	
22	
23	
24	
25	

Description

1 vv) *DesignMode*

2 ww) *DetectUrls*

3 xx) *UseStdAccessibleObjects*

4
5
6 *Description*

7 Gets or sets a value indicating whether or not the
8 **System.Windows.Forms.RichTextBox** will automatically format a Uniform
9 Resource Locator (URL) when it is typed into the control.

10 If this property is set to **true** , any text entered into the control that is
11 determined by the **System.Windows.Forms.RichTextBox** to be a URL is
12 automatically formatted as a link. You can create an event handler for the
13 **System.Windows.Forms.RichTextBox.LinkClicked** event to handle all links
14 clicked in the control. The **System.Windows.Forms.LinkClickedEventArgs**
15 that is provided to the event handler for the
16 **System.Windows.Forms.RichTextBox.LinkClicked** event provides data that
17 enables you to determine which link was clicked in the control in order to process
18 the link.

14 yy) *DisplayRectangle*

15 zz) *Disposing*

16 aaa) *Dock*

17 bbb) *Enabled*

18 ccc) *Events*

19 ddd) *Focused*

20 eee) *Font*

21 fff) *UseStdAccessibleObjects*

22
23
24
25 *Description*

Gets or sets the font used when displaying text in the control.

ggg) FontHeight

hhh) ForeColor

iii) UseStdAccessibleObjects

Description

Gets or sets the font used when displaying text in the control.

MSI-861US.APP

- jjj) Handle*
- kkk) HasChildren*
- lll) Height*
- mmm) HideSelection*
- nnn) ImeMode*
- ooo) InvokeRequired*
- ppp) IsAccessible*
- qqq) IsDisposed*
- rrr) IsHandleCreated*
- sss) Left*
- ttt) Lines*
- uuu) Location*
- vvv) MaxLength*
- www) UseStdAccessibleObjects*

Description

xxx) *Modified*

yyy) *Multiline*

zzz) *UseStdAccessibleObjects*

aaaa) *Name*

bbbb) *Parent*

cccc) *PreferredHeight*

dddd) *ProductName*

eeee) *ProductVersion*

ffff) *ReadOnly*

gggg) *RecreatingHandle*

hhhh) *RedoActionName*

iiii) *UseStdAccessibleObjects*

Description

Gets the name of the action that can be reapplied to the control when the **System.Windows.Forms.RichTextBox.Redo** method is called.

If this property returns an empty string (""), there is no operation available to reapply to the control. You can use this method to determine the last action undone in the **System.Windows.Forms.RichTextBox** control that can then be reapplied to the control when a call to the **System.Windows.Forms.RichTextBox.Redo** method is made. You can determine whether there are any operations to be reapplied to the control by using the **System.Windows.Forms.RichTextBox.CanRedo** property.

jjjj) Region

kkkk) RenderRightToLeft

llll) ResizeRedraw

mmmm)Right

nnnn) RightMargin

oooo) UseStdAccessibleObjects

Description

Gets or sets the size of a single line of text within the **System.Windows.Forms.RichTextBox** control.

When a value greater than zero is entered into the control, a nonvisible margin is placed in the control at the specified number of pixels from the left side of the control. Any text that is entered that extends beyond this margin is placed on the next line of text in the control. This property affects all text currently entered into the control as well as any additional text entered into the control after the property is set. You can use this property to specify a maximum line width for all text entered into a **System.Windows.Forms.RichTextBox** control.

pppp) RightToLeft

qqqq) Rtf

rrrr) UseStdAccessibleObjects

Description

Gets or sets the text of the **System.Windows.Forms.RichTextBox** control, including all Rich Text Format (RTF) codes.

You can use this property to place RTF formatted text into the control for display or to extract the text of the control with the specified RTF formatting defined in the text of the control. This property is typically used when you are assigning

RTF text from another RTF source, such as Microsoft Word or Windows WordPad, to the control.

ssss) ScrollBars

tttt) UseStdAccessibleObjects

[C#] public RichTextBoxScrollBars ScrollBars {get; set;}

[C++] public: __property RichTextBoxScrollBars get_ScrollBars();public:

__property void set_ScrollBars(RichTextBoxScrollBars);

[VB] Public Property ScrollBars As RichTextBoxScrollBars

[JScript] public function get ScrollBars() : RichTextBoxScrollBars;public function

set ScrollBars(RichTextBoxScrollBars);

Description

Gets or sets the type of scroll bars to display in the **System.Windows.Forms.RichTextBox** control.

This property enables you to provide horizontal and vertical scroll bars to the user of the **System.Windows.Forms.RichTextBox** control to enable scrolling text within the control that is outside of the physical dimensions of the control. You can also use this property to remove scroll bars from the control to restrict scrolling the contents of the control.

uuuu) SelectedRtf

vvvv) UseStdAccessibleObjects

[C#] public string SelectedRtf {get; set;}

[C++] public: __property String* get_SelectedRtf();public: __property void

set_SelectedRtf(String*);

[VB] Public Property SelectedRtf As String

1 [JScript] public function get SelectedRtf() : String;public function set
2 SelectedRtf(String);

3
4 *Description*

5 Gets or sets the currently selected Rich Text Format (RTF) formatted text in the
6 control.

7 This property enables you to obtain the selected text in the control, including the
8 RTF formatting codes. You can use this property to copy text from your control,
9 complete with formatting, and paste the text in other applications that accept
10 RTF formatted text, such as Microsoft Word and Windows WordPad. To get the
11 selected text, without RTF formatting codes, use the
12 **System.Windows.Forms.TextBoxBase.SelectedText** property.

13 *www.SelectedText*

14 *xxxx) UseStdAccessibleObjects*

15 [C#] public override string SelectedText {get; set;}

16 [C++] public: __property virtual String* get_SelectedText();public: __property
17 virtual void set_SelectedText(String*);

18 [VB] Overrides Public Property SelectedText As String

19 [JScript] public function get SelectedText() : String;public function set
20 SelectedText(String);

21
22 *Description*

23 Gets or sets the selected text within the
24 **System.Windows.Forms.RichTextBox** .
25

yyyy) *SelectionAlignment*

zzzz) *UseStdAccessibleObjects*

[C#] public HorizontalAlignment SelectionAlignment {get; set;}

[C++] public: __property HorizontalAlignment get_SelectionAlignment();public:

__property void set_SelectionAlignment(HorizontalAlignment);

[VB] Public Property SelectionAlignment As HorizontalAlignment

[JScript] public function get SelectionAlignment() : HorizontalAlignment;public

function set SelectionAlignment(HorizontalAlignment);

Description

Gets or sets the alignment to apply to the current selection or insertion point.

If no paragraph is selected in the control, setting this property applies the alignment setting to the paragraph in which the insertion point appears as well as to paragraphs created after the paragraph that has the alignment property setting. For example, if there are two paragraphs in a **System.Windows.Forms.RichTextBox** control and the insertion point is located within the second paragraph. If you set the **System.Windows.Forms.RichTextBox.SelectionAlignment** property to **HorizontalAlignment.Center**, the paragraph at the insertion point will be centered within the control. If a third paragraph is created after the second paragraph, it also is aligned to the center of the control.

aaaaa) *SelectionBullet*

bbbbbb) *UseStdAccessibleObjects*

[C#] public bool SelectionBullet {get; set;}

[C++] public: __property bool get_SelectionBullet();public: __property void

set_SelectionBullet(bool);

[VB] Public Property SelectionBullet As Boolean

1 [JScript] public function get SelectionBullet() : Boolean;public function set
2 SelectionBullet(Boolean);
3

4 *Description*

5 Gets or sets a value indicating whether the bullet style is applied to the current
6 selection or insertion point.

7 If no text is selected, the bullet style is applied to the current insertion point and
8 to all paragraphs that the user enters after the insertion point. The bullet style is
9 applied to the text of the control until the insertion point is moved or when the
10 user presses the Enter key on an empty bullet item.

11 *cccccc) SelectionCharOffset*

12 *dddddd) UseStdAccessibleObjects*

13 [C#] public int SelectionCharOffset {get; set;}

14 [C++] public: __property int get_SelectionCharOffset();public: __property void
15 set_SelectionCharOffset(int);

16 [VB] Public Property SelectionCharOffset As Integer

17 [JScript] public function get SelectionCharOffset() : int;public function set
18 SelectionCharOffset(int);
19

20 *Description*

21 Gets or sets whether text in the control appears on the baseline, as a
22 superscript, or as a subscript below the baseline.

23 If this property is set to zero, the text appears on the baseline. If it is a positive
24 number, the number specifies the number of pixels by which to raise the text
25 selection above the baseline. If it is a negative number, this number specifies the
number of pixels by which to subscript the text selection. You can use this
property to specify text as superscript or subscript.

eeee) SelectionColor

ffff) UseStdAccessibleObjects

[C#] public Color SelectionColor {get; set;}

[C++] public: __property Color get_SelectionColor();public: __property void
set_SelectionColor(Color);

[VB] Public Property SelectionColor As Color

[JScript] public function get SelectionColor() : Color;public function set
SelectionColor(Color);

Description

Gets or sets the text color of the current text selection or insertion point.

If the current text selection has more than one color specified, this property returns **Color.Empty** . If no text is currently selected, the text color specified in this property is applied to the current insertion point and to all text that is typed into the control after the insertion point. The text color setting applies until the property is changed to a different color or until the insertion point is moved to a different section within the control.

ggggg) SelectionFont

hhhhh)UseStdAccessibleObjects

[C#] public Font SelectionFont {get; set;}

[C++] public: __property Font* get_SelectionFont();public: __property void
set_SelectionFont(Font*);

[VB] Public Property SelectionFont As Font

[JScript] public function get SelectionFont() : Font;public function set
SelectionFont(Font);

Description

Gets or sets the font of the current text selection or insertion point.

If the current text selection has more than one font specified, this property is **null**. If no text is currently selected, the font specified in this property is applied to the current insertion point and to all text that is typed into the control after the insertion point. The font setting applies until the property is changed to a different font or until the insertion point is moved to a different section within the control.

iiii) SelectionHangingIndent

jjjj) UseStdAccessibleObjects

```
[C#]      public      int      SelectionHangingIndent      {get;      set;}
```

```
[C++] public: __property int get_SelectionHangingIndent();public: __property  
void      set_SelectionHangingIndent(int);
```

```
[VB]      Public      Property      SelectionHangingIndent      As      Integer
```

```
[JScript] public function get SelectionHangingIndent() : int;public function set  
SelectionHangingIndent(int);
```

Description

Gets or sets the distance between the left edge of the first line of text in the selected paragraph and the left edge of subsequent lines in the same paragraph.

If no text is currently selected, the hanging indent is applied to the paragraph in which the insertion point appears and to all text that is typed into the control after the insertion point. The hanging indent setting applies until the property is changed to a different value or until the insertion point is moved to a different paragraph within the control.

IIII) *UseStdAccessibleObjects*

Description

If no text is currently selected, the indentation setting is applied to the paragraph in which the insertion point appears and to all text that is typed into the control after the insertion point. The indentation setting applies until the property is changed to a different value or until the insertion point is moved to a different paragraph within the control.

nnnnn)UseStdAccessibleObjects

```
[C#]      public      override      int      SelectionLength      {get;      set;}

[C++] public: __property virtual int get_SelectionLength();public: __property
virtual      void      set_SelectionLength(int);

[VB]      Overrides      Public      Property      SelectionLength      As      Integer

[JScript] public function get SelectionLength() : int;public function set
SelectionLength(int);
```

Description

Gets or sets the number of characters selected in control.

You can use this property to determine if any characters are currently selected in the text box control before performing operations on the selected text. You can also use this property to determine the total number of characters (including spaces) that are selected when performing single character tasks in a **for** loop.

ooooo) SelectionProtected

ppppp) UseStdAccessibleObjects

[C#] public bool SelectionProtected {get; set;}

[C++] public: __property bool get_SelectionProtected();public: __property void set_SelectionProtected(bool);

[VB] Public Property SelectionProtected As Boolean

[JScript] public function get SelectionProtected() : Boolean;public function set SelectionProtected(Boolean);

Description

Gets or sets a value indicating whether the current text selection is protected.

If no text is currently selected, the protection setting is applied to the paragraph in which the insertion point appears and to all text that is typed into the control after the insertion point. The protection setting applies until the property is changed to a different value or until the insertion point is moved to a different paragraph within the control.

qqqqq) SelectionRightIndent

rrrrr) UseStdAccessibleObjects

[C#] public int SelectionRightIndent {get; set;}

1 [C++] public: __property int get_SelectionRightIndent();public: __property void
2 set_SelectionRightIndent(int);

3 [VB] Public Property SelectionRightIndent As Integer

4 [JScript] public function get SelectionRightIndent() : int;public function set
5 SelectionRightIndent(int);

6 7 *Description*

8 The distance (in pixels) between the right edge of the RichTextBox control and
the right edge of the text that is selected or added at the current insertion point.

9 If no text is currently selected, the indentation setting is applied to the
10 paragraph in which the insertion point appears and to all text that is typed into
the control after the insertion point. The indentation setting applies until the
11 property is changed to a different value or until the insertion point is moved to a
different paragraph within the control.

12
13 *sssss) SelectionStart*

14 *ttttt) SelectionTabs*

15 *uuuuu) UseStdAccessibleObjects*

16 17 18 *Description*

19 Gets or sets the absolute tab stop positions in a
System.Windows.Forms.RichTextBox control.

20 This property enables you to obtain an array that contains the spacing for each
21 tab in the current text selection within the
System.Windows.Forms.RichTextBox control. You can then use this
22 property to adjust the size of each tab within the text selection. For example, if
you want to adjust the tab space within the document, you can select the entire
23 document and obtain the list of tab spaces using the
System.Windows.Forms.RichTextBox.SelectionTabs property. You can
24 then adjust them to new values and reassign them to this property.

vvvvv) *SelectionType*

wwwww) *UseStdAccessibleObjects*

```
[C#]    public    RichTextBoxSelectionTypes    SelectionType    {get;}
[C++]  public:  __property RichTextBoxSelectionTypes  get_SelectionType();
[VB]   Public ReadOnly Property SelectionType As RichTextBoxSelectionTypes
[JScript] public function get SelectionType() : RichTextBoxSelectionTypes;
```

Description

Gets the selection type within the control.

You can use this property to determine the type of data that is currently selected in the control in order to handle the selection properly when performing tasks within the control on the current selection. The property can represent any combination of values from the

System.Windows.Forms.RichTextBoxSelectionTypes enumeration representing the many types of objects in the current selection.

xxxxx) *ShowFocusCues*

yyyyy) *ShowKeyboardCues*

zzzzz) *ShowSelectionMargin*

aaaaaa) *UseStdAccessibleObjects*

Description

Gets or sets a value indicating whether a selection margin is displayed in the **System.Windows.Forms.RichTextBox**.

You can use this property to enable the user to easily select lines of text in the **System.Windows.Forms.RichTextBox**. The selection margin is added to the left side of the **System.Windows.Forms.RichTextBox**. This margin makes it easier for the user to select text starting on the left side of the control. The user

can click in the selection margin to select a single line of text or double-click to select the entire paragraph that the line double-clicked is contained within.

bbbbbb)Site

cccccc)Size

dddddd)TabIndex

eeeeee)TabStop

ffffff) Tag

gggggg)Text

hhhhh)UseStdAccessibleObjects

iiiiii) TextLength

jjjjjj) Top

kkkkkk)TopLevelControl

lllll) UndoActionName

mmmmmm)UseStdAccessibleObjects

Description

Gets the name of the action that can be undone in the control when the **System.Windows.Forms.TextBoxBase.Undo** method is called.

This property enables you to determine the last action that was done within the control that can be undone. You can use this property to limit the operations available to be undone by the user of the control.

1 *nnnnnn)Visible*

2 *ooooooo)Width*

3 *ppppppp)WindowTarget*

4 *qqqqqqq)WordWrap*

5 *rrrrrrr) ZoomFactor*

6 *sssssss) UseStdAccessibleObjects*

7
8
9 *Description*

10 Gets or sets the current zoom level of the
11 **System.Windows.Forms.RichTextBox** .

12 The value of this property can be between 0.64 and 64.0. A value of 1.0
13 indicates that no zoom is applied to the control. The zoom feature performs
14 optimally when the document contains TrueType fonts. When a non-TrueType
15 font is used within the document of the control, the
16 **System.Windows.Forms.RichTextBox.ZoomFactor** property will use the
17 nearest whole number value. You can use this property to enable the user of the
18 **System.Windows.Forms.RichTextBox** control to zoom into sections of the
19 documentation that are too small to view or to condense the view to enable
20 more of the document to be viewed on screen.

17 *ttttt) UseStdAccessibleObjects*

20 *Description*

21 Occurs when contents within the control are resized.

22 For more information about handling events, see .

1 uuuuuu)UseStdAccessibleObjects

2 vvvvvv)UseStdAccessibleObjects

3
4
5 *Description*

6 Occurs when the user completes a drag-and-drop

7 wwwwww)UseStdAccessibleObjects

8
9 [C#] public new event DragEventHandler DragEnter;

10 [C++] public: __event DragEventHandler* DragEnter;

11 [VB] Shadows Public Event DragEnter As DragEventHandler

12
13 *Description*

14
15 xxxxxx)UseStdAccessibleObjects

16
17 [C#] public new event EventHandler DragLeave;

18 [C++] public: __event EventHandler* DragLeave;

19 [VB] Shadows Public Event DragLeave As EventHandler

20
21 *Description*

22
23 yyyyyy)UseStdAccessibleObjects

24
25 [C#] public new event DragEventHandler DragOver;

```

1  [C++]      public:      __event      DragEventHandler*      DragOver;
2  [VB]      Shadows      Public      Event      DragOver      As      DragEventHandler
3

```

Description

zzzzzz) UseStdAccessibleObjects

Description

aaaaaaa) UseStdAccessibleObjects

Description

Occurs when the user clicks the horizontal scroll bar of the control.

For more information about handling events, see .

bbbbbbb) UseStdAccessibleObjects

```

19 [C#]      public      event      EventHandler      ImeChange;

```

```

20 [C++]      public:      __event      EventHandler*      ImeChange;

```

```

21 [VB]      Public      Event      ImeChange      As      EventHandler

```

Description

Occurs when the user switches input methods on an Asian version of the Windows operating system.

For more information about handling events, see .

ccccccc)UseStdAccessibleObjects

Description

Occurs when the user clicks on a link within the text of the control.

You can create an event handler for this event to process a link that has been clicked within the control. Using the information provided to the event handler, you can determine which link was clicked in the document.

ddddddd)UseStdAccessibleObjects

Description

Occurs when the user attempts to modify protected text in the control.

You can create an event handler for this event in your applications to determine when the user has attempted to modify text that has been marked as protected in the control. The event handler can be used to notify the user that the text the user is attempting to modify is protected or to display a dialog box that enables the user to make appropriate changes to the text. For example, if the protected area is a date, you can display a dialog box that enables the user to choose a date which can then be applied to the text of the control.

eeeeeee)UseStdAccessibleObjects

Description

ffffff) UseStdAccessibleObjects

Description

Occurs when the selection of text within the control has changed.

You can create an event handler for this event to determine when the user has changed text selection within the control. An event handler for this event can be used to keep text selected until the user has completed a task within the application.

ggggggg) UseStdAccessibleObjects

Description

Occurs when the user clicks the vertical scroll bars of the control.

For more information about handling events, see .

hhhhhhh) CanPaste

[C#] public bool CanPaste(DataFormats.Format clipFormat);

[C++] public: bool CanPaste(DataFormats.Format* clipFormat);

[VB] Public Function CanPaste(ByVal clipFormat As DataFormats.Format) As

Boolean

[JScript] public function CanPaste(clipFormat : DataFormats.Format) : Boolean;

Description

Determines whether you can paste information from the Clipboard in the specified data format.

Return Value: **true** if you can paste data from the Clipboard in the specified data format; otherwise, **false** .

You can use this method to determine whether the current contents of the Clipboard are in a specified Clipboard data format before allowing the user to paste the information into the **System.Windows.Forms.RichTextBox** control. For example, you could create an event handler for a **System.Windows.Forms.MenuItem.Popup** event of a paste command **System.Windows.Forms.MenuItem** and use this method to determine whether the paste **System.Windows.Forms.MenuItem** should be enabled based on the type of data in the Clipboard. One of the **System.Windows.Forms.DataFormats.Format** values.

iiiiiii) CreateRichEditOleCallback

```
[C#]      protected      virtual      object      CreateRichEditOleCallback();
[C++]      protected:      virtual      Object*      CreateRichEditOleCallback();
[VB]      Overridable Protected Function CreateRichEditOleCallback() As Object
[JScript]      protected      function      CreateRichEditOleCallback()      :      Object;
```

Description

Creates an **IRichEditOleCallback** compatible object for handling RichEdit callback operations.

Return Value: An object that implements the **IRichEditOleCallback** interface.

You can override this method in your derived class to allow access to the underlying RichEdit features. If you override this method, all drag and drop events will not be raised. You will have to provide your own support for drag and drop as a result. For more information on the **IRichEditOleCallback** interface, refer to the MSDN documentation in the Platform SDK reference.

jjjjjj) Find

```
[C#]      public      int      Find(char[]      characterSet);
[C++]      public:      int      Find(__wchar_t      characterSet      __gc[]);
[VB]      Public Function Find(ByVal characterSet() As Char) As Integer
[JScript]      public      function      Find(characterSet      :      Char[])      :      int;
```

1
2 *Description*

3 Searches the text of a **System.Windows.Forms.RichTextBox** control for the first instance of a character from a list of characters.

4 *Return Value:* The location within the control where the search characters were found or a negative one (-1) if the search characters are not found or an empty search character set is specified in the *char* parameter.

6 This version of the

7 **System.Windows.Forms.RichTextBox.Find(System.String)** method searches for the first instance of a character from a list of characters specified in the *characterSet* parameter and returns the location of the character. For example, you pass an array of characters containing the character 'Q'. If the control contained the text "The Quick Brown Fox", the **System.Windows.Forms.RichTextBox.Find(System.String)** method would return the value of four. An upper case character and a lower case character are considered different values in the search. The array of characters to search for.

11 *kkkkkkk)Find*

12
13
14 [C#] public int Find(string str);
15 [C++] public: int Find(String* str);
16 [VB] Public Function Find(ByVal str As String) As Integer
17 [JScript] public function Find(str : String) : int; Searches for text within the
18 contents of the **System.Windows.Forms.RichTextBox** .

19
20 *Description*

21 Searches the text in a **System.Windows.Forms.RichTextBox** control for a string.

22 *Return Value:* The location within the control where the search text was found or a negative one (-1) if the search string is not found or an empty search string is specified in the *str* parameter.

23
24 The **System.Windows.Forms.RichTextBox.Find(System.String)** method searches for the text specified in the *str* parameter and returns the location of the first character within the control. If the property returns a negative value, the
25

text string being searched for was not found within the contents of the control. You can use this method to create search functionality that can be provided to the user of the control. You can also use this method to search for text to be replaced with a specific format. For example, if the user entered dates into the control, you could use the

System.Windows.Forms.RichTextBox.Find(System.String) method to search for all dates in the document and replace them with the appropriate format before using the **System.Windows.Forms.RichTextBox.SaveFile(System.String)** method of the control. The text to locate in the control.

IIIIII) Find

[C#] public int Find(char[] characterSet, int start);

[C++] public: int Find(__wchar_t characterSet __gc[], int start);

[VB] Public Function Find(ByVal characterSet() As Char, ByVal start As Integer)

As Integer

[JScript] public function Find(characterSet : Char[], start : int) : int;

Description

Searches the text of a **System.Windows.Forms.RichTextBox** control, at a specific starting point, for the first instance of a character from a list of characters.

Return Value: The location within the control where the search characters are found.

This version of the

System.Windows.Forms.RichTextBox.Find(System.String) method searches for the first instance of a character from a list of characters specified in the *characterSet* parameter and returns the location the character. For example, you pass an array of characters containing the character 'Q'. If the control contained the text "The Quick Brown Fox", the

System.Windows.Forms.RichTextBox.Find(System.String) method would return the value of four. An upper case character and a lower case character are considered different values in the search. The array of characters to search for. The location within the control's text at which to begin searching.

mmmmmmmm)Find

```
[C#]    public    int    Find(string    str,    RichTextBoxFinds    options);
[C++]    public:    int    Find(String*    str,    RichTextBoxFinds    options);
[VB]    Public Function Find(ByVal str As String, ByVal options As
RichTextBoxFinds)                                As                                Integer
[JScript] public function Find(str : String, options : RichTextBoxFinds) : int;
```

Description

Searches the text in a **System.Windows.Forms.RichTextBox** control for a string with specific options applied to the search.

Return Value: The location within the control where the search text was found.

The **System.Windows.Forms.RichTextBox.Find(System.String)** method searches for the text specified in the *str* parameter and returns the location of the first character within the control. If the property returns a negative value, the text string being searched for was not found within the contents of the control. You can use this method to create search functionality that can be provided to the user of the control. You can also use this method to search for text to be replaced with a specific format. For example, if the user entered dates into the control, you can use the

System.Windows.Forms.RichTextBox.Find(System.String) method to search for all dates in the document and replace them with the appropriate format before using the

System.Windows.Forms.RichTextBox.SaveFile(System.String) method of the control. The text to locate in the control. A bitwise combination of the **System.Windows.Forms.RichTextBoxFinds** values.

nnnnnnnn)Find

```
[C#]    public    int    Find(char[]    characterSet,    int    start,    int    end);
[C++]    public:    int    Find(__wchar_t    characterSet    __gc[],    int    start,    int    end);
[VB]    Public Function Find(ByVal characterSet() As Char, ByVal start As Integer,
ByVal                                end                                As                                Integer)                                As                                Integer
```

1 [JScript] public function Find(characterSet : Char[], start : int, end : int) : int;

2
3 *Description*

4 Searches a range of text in a **System.Windows.Forms.RichTextBox** control
5 for the first instance of a character from a list of characters.

6 *Return Value:* The location within the control where the search characters are
7 found.

8 This version of the

9 **System.Windows.Forms.RichTextBox.Find(System.String)** method
10 searches for the first instance of a character from a list of characters specified in
11 the *characterSet* parameter and returns the location of the character. For
12 example, you pass an array of characters containing the character 'Q'. If the
13 control contained the text "The Quick Brown Fox", the

14 **System.Windows.Forms.RichTextBox.Find(System.String)** method would
15 return the value of four. An upper case character and a lower case character are
16 considered different values in the search. The array of characters to search for.
17 The location within the control's text at which to begin searching. The location
18 within the control's text at which to end searching.

19 *ooooooo)Find*

20 [C#] public int Find(string str, int start, RichTextBoxFinds options);

21 [C++] public: int Find(String* str, int start, RichTextBoxFinds options);

22 [VB] Public Function Find(ByVal str As String, ByVal start As Integer, ByVal
23 options As RichTextBoxFinds) As Integer

24 [JScript] public function Find(str : String, start : int, options : RichTextBoxFinds) :
25 int;

26
27 *Description*

28 Searches the text in a **System.Windows.Forms.RichTextBox** control for a
29 string at a specific location within the control and with specific options applied to
30 the search.

31 *Return Value:* The location within the control where the search text was found.

The **System.Windows.Forms.RichTextBox.Find(System.String)** method searches for the text specified in the *str* parameter and returns the location of the first character of the search string within the control. If the property returns a negative value, the text string being searched for was not found within the contents of the control. You can use this method to create search functionality that can be provided to the user of the control. You can also use this method to search for text to be replaced with a specific format. For example, if the user entered dates into the control, you could use the **System.Windows.Forms.RichTextBox.Find(System.String)** method to search for all dates in the document and replace them with the appropriate format before using the **System.Windows.Forms.RichTextBox.SaveFile(System.String)** method of the control. The text to locate in the control. The location within the control's text at which to begin searching. A bitwise combination of the **System.Windows.Forms.RichTextBoxFinds** values.

ppppppp)Find

```
[C#] public int Find(string str, int start, int end, RichTextBoxFinds options);
[C++] public: int Find(String* str, int start, int end, RichTextBoxFinds options);
[VB] Public Function Find(ByVal str As String, ByVal start As Integer, ByVal
end As Integer, ByVal options As RichTextBoxFinds) As Integer
[JavaScript] public function Find(str : String, start : int, end : int, options :
RichTextBoxFinds) : int;
```

Description

Searches the text in a **System.Windows.Forms.RichTextBox** control for a string within a range of text within the control and with specific options applied to the search.

Return Value: The location within the control where the search text was found.

The **System.Windows.Forms.RichTextBox.Find(System.String)** method searches for the text specified in the *str* parameter and returns the location of the first character of the search string within the control. If the property returns a negative value, the text string being searched for was not found within the contents of the control. You can use this method to create search functionality that can be provided to the user of the control. You can also use this method to

search for text to be replaced with a specific format. For example, if the user entered dates into the control, you can use the

System.Windows.Forms.RichTextBox.Find(System.String) method to search for all dates in the document and replace them with the appropriate format before using the

System.Windows.Forms.RichTextBox.SaveFile(System.String) method of the control. The text to locate in the control. The location within the control's text at which to begin searching. The location within the control's text at which to end searching. This value must be equal to negative one (-1) or greater than or equal to the *start* parameter. A bitwise combination of the **System.Windows.Forms.RichTextBoxFinds** values.

qqqqqq)GetCharFromPosition

[C#] public char GetCharFromPosition(Point pt);

[C++] public: __wchar_t GetCharFromPosition(Point pt);

[VB] Public Function GetCharFromPosition(ByVal pt As Point) As Char

[JScript] public function GetCharFromPosition(pt : Point) : Char;

Description

Gets the character that is closest to the specified location within the control.

Return Value: The character at the specified location.

If the location specified in the *pt* parameter is outside the client area of the control, the first character of the string closest to the point specified in *pt* is returned. You can use this method to determine which characters are located near a specific point within the control. You can then use this value to perform operations on the text at that location. The location from which to seek the nearest character.

rrrrrr)GetCharIndexFromPosition

[C#] public int GetCharIndexFromPosition(Point pt);

[C++] public: int GetCharIndexFromPosition(Point pt);

[VB] Public Function GetCharIndexFromPosition(ByVal pt As Point) As Integer

1 [JScript] public function GetCharIndexFromPosition(pt : Point) : int;

3 *Description*

4 Gets the index of the character nearest to the specified location.

Return Value: The zero-based character index at the specified location.

5 This method returns the character index that is closest to the position specified
6 in the *pt* parameter. The character index is a zero-based index of text in the
7 control, including spaces. You can use this method to determine where in the
8 text the user has the mouse over by passing the mouse coordinates to this
9 method. This can be useful if you want to perform tasks when the user hovers
10 the mouse pointer over a word in the text of the control. The location to search.

9 *ssssss)GetLineFromCharIndex*

11 [C#] public int GetLineFromCharIndex(int index);

12 [C++] public: int GetLineFromCharIndex(int index);

13 [VB] Public Function GetLineFromCharIndex(ByVal index As Integer) As Integer

14 [JScript] public function GetLineFromCharIndex(index : int) : int;

16 *Description*

17 Gets the line number from the specified character position within the text of the
18 **System.Windows.Forms.RichTextBox** control.

Return Value: The zero-based line number where the character index is located
19 in.

20 This method enables you to determine the line number based on the character
21 index specified in the *index* parameter of the method. The first line of text in the
22 control returns the value zero. You can use this method to determine which line
23 a specific character index is located within. For example, after calling the
24 **System.Windows.Forms.RichTextBox.Find(System.String)** method to
25 search for text, you can obtain the character index to where the search results
are found. You can call this method with the character index returned by the
System.Windows.Forms.RichTextBox.Find(System.String) method to
determine which line the word was found. The character index position to
search.

GetPositionFromCharIndex

[C#] public Point GetPositionFromCharIndex(int index);

[C++] public: Point GetPositionFromCharIndex(int index);

[VB] Public Function GetPositionFromCharIndex(ByVal index As Integer) As Point

[JScript] public function GetPositionFromCharIndex(index : int) : Point;

Description

Gets the location within the control at the specified character index.

Return Value: The location of the specified character.

This method enables you to determine where in the control a specific character index is located. You can use this method for such tasks as displaying context menu items or help information for a word in the control. For example, if you wanted to display a menu of options to the user when the user right clicks on a word in the control, you can use this method to determine the position of the word to properly display a **System.Windows.Forms.ContextMenu** control. The index of the character for which to retrieve the location.

LoadFile

[C#] public void LoadFile(string path);

[C++] public: void LoadFile(String* path);

[VB] Public Sub LoadFile(ByVal path As String)

[JScript] public function LoadFile(path : String); Loads the contents of a file into the **System.Windows.Forms.RichTextBox** control.

Description

Loads a Rich Text Format (RTF) or standard ASCII text file into the **System.Windows.Forms.RichTextBox** control.

When loading a file with the

System.Windows.Forms.RichTextBox.LoadFile(System.String) method, the contents of the file being loaded replace the entire contents of the **System.Windows.Forms.RichTextBox** control. This will cause the values of the **System.Windows.Forms.TextBoxBase.Text** and **System.Windows.Forms.RichTextBox.Rtf** properties to change. You can use this method to load a previously created text or RTF document into the control for manipulation. If you want to save the file, you can use the **System.Windows.Forms.RichTextBox.SaveFile(System.String)** method. The name and location of the file to load into the control.

vvvvvvv)LoadFile

[C#] public void LoadFile(Stream data, RichTextBoxStreamType fileType);

[C++] public: void LoadFile(Stream* data, RichTextBoxStreamType fileType);

[VB] Public Sub LoadFile(ByVal data As Stream, ByVal fileType As RichTextBoxStreamType)

[JScript] public function LoadFile(data : Stream, fileType : RichTextBoxStreamType);

Description

Loads the contents of an existing data stream into the **System.Windows.Forms.RichTextBox** control.

You can use this version of the

System.Windows.Forms.RichTextBox.LoadFile(System.String) method to load the **System.Windows.Forms.RichTextBox** with data from an existing stream of data. The data that is loaded into the control replaces the entire contents of the **System.Windows.Forms.RichTextBox** control. This will cause the values of the **System.Windows.Forms.TextBoxBase.Text** and **System.Windows.Forms.RichTextBox.Rtf** properties to change. You can use this method to load a file that has been previously opened into a data stream into the control for manipulation. If you want to save contents of the control back into the stream, you can use the **System.Windows.Forms.RichTextBox.SaveFile(System.String)** method that accepts a **System.IO.Stream** object as a parameter. A stream of data to

load into the **System.Windows.Forms.RichTextBox** control. One of the **System.Windows.Forms.RichTextBoxStreamType** values.

wwwwww)LoadFile

```
[C#] public void LoadFile(string path, RichTextBoxStreamType fileType);  
[C++] public: void LoadFile(String* path, RichTextBoxStreamType fileType);  
[VB] Public Sub LoadFile(ByVal path As String, ByVal fileType As  
RichTextBoxStreamType)  
[JScript] public function LoadFile(path : String, fileType :  
RichTextBoxStreamType);
```

Description

Loads a specific type of file into the **System.Windows.Forms.RichTextBox** control.

When loading a file with the **System.Windows.Forms.RichTextBox.LoadFile(System.String)** method, the contents of the file being loaded replace the entire contents of the **System.Windows.Forms.RichTextBox** control. This will cause the values of the **System.Windows.Forms.TextBoxBase.Text** and **System.Windows.Forms.RichTextBox.Rtf** properties to change. You can use this method to load a previously created text or Rich Text Format (RTF) document into the control for manipulation. If you want to save the file, you can use the **System.Windows.Forms.RichTextBox.SaveFile(System.String)** method. The name and location of the file to load into the control. One of the **System.Windows.Forms.RichTextBoxStreamType** values.

xxxxxxx)OnBackColorChanged

```
[C#] protected override void OnBackColorChanged(EventArgs e);  
[C++] protected: void OnBackColorChanged(EventArgs* e);
```

1 [VB] Overrides Protected Sub OnBackColorChanged(ByVal e As EventArgs)

2 [JScript] protected override function OnBackColorChanged(e : EventArgs);

3 *yyyyyyy)OnContentsResized*

5 [C#] protected virtual void OnContentsResized(ContentsResizedEventArgs e);

6 [C++] protected: virtual void OnContentsResized(ContentsResizedEventArgs* e);

7 [VB] Overridable Protected Sub OnContentsResized(ByVal e As
8 ContentsResizedEventArgs)

9 [JScript] protected function OnContentsResized(e : ContentsResizedEventArgs);

11 *Description*

12 Raises the **System.Windows.Forms.RichTextBox.ContentsResized** event.

13 Raising an event invokes the event handler through a delegate. For more
14 information, see . A **System.Windows.Forms.ContentsResizedEventArgs**
that contains the event data.

15 *zzzzzzz)OnContextMenuChanged*

17 [C#] protected override void OnContextMenuChanged(EventArgs e);

18 [C++] protected: void OnContextMenuChanged(EventArgs* e);

19 [VB] Overrides Protected Sub OnContextMenuChanged(ByVal e As EventArgs)

20 [JScript] protected override function OnContextMenuChanged(e : EventArgs);

22 *Description*

aaaaaaa)OnHandleCreated

```
[C#]    protected    override    void    OnHandleCreated(EventArgs    e);
[C++]    protected:    void    OnHandleCreated(EventArgs*    e);
[VB]    Overrides Protected Sub OnHandleCreated(ByVal e As EventArgs)
[JScript] protected override function OnHandleCreated(e : EventArgs);
```

bbbbbbb)OnHandleDestroyed

```
[C#]    protected    override    void    OnHandleDestroyed(EventArgs    e);
[C++]    protected:    void    OnHandleDestroyed(EventArgs*    e);
[VB]    Overrides Protected Sub OnHandleDestroyed(ByVal e As EventArgs)
[JScript] protected override function OnHandleDestroyed(e : EventArgs);
```

ccccccc)OnHScroll

```
[C#]    protected    virtual    void    OnHScroll(EventArgs    e);
[C++]    protected:    virtual    void    OnHScroll(EventArgs*    e);
[VB]    Overridable Protected Sub OnHScroll(ByVal e As EventArgs)
[JScript]    protected    function    OnHScroll(e    :    EventArgs);
```

Description

Raises the **System.Windows.Forms.RichTextBox.HScroll** event.

Raising an event invokes the event handler through a delegate. For more information, see . An **System.EventArgs** that contains the event data.

ddddddd)OnImeChange

```
1
2
3 [C#]    protected    virtual    void    OnImeChange(EventArgs    e);
4 [C++]    protected:    virtual    void    OnImeChange(EventArgs*    e);
5 [VB]    Overridable    Protected    Sub    OnImeChange(ByVal    e As    EventArgs)
6 [JScript]    protected    function    OnImeChange(e    :    EventArgs);
```

Description

Raises the **System.Windows.Forms.RichTextBox.ImeChange** event.

Raising an event invokes the event handler through a delegate. For more information, see . An **System.EventArgs** that contains the event data.

eeeeeee)OnLinkClicked

```
12
13 [C#]    protected    virtual    void    OnLinkClicked(LinkClickedEventArgs    e);
14 [C++]    protected:    virtual    void    OnLinkClicked(LinkClickedEventArgs*    e);
15 [VB]    Overridable    Protected    Sub    OnLinkClicked(ByVal    e As
16 LinkClickedEventArgs)
17 [JScript]    protected    function    OnLinkClicked(e    :    LinkClickedEventArgs);
18
```

Description

Raises the **System.Windows.Forms.RichTextBox.LinkClicked** event.

Raising an event invokes the event handler through a delegate. For more information, see . A **System.Windows.Forms.LinkClickedEventArgs** that contains the event data.

ffffff)OnProtected

```
[C#]    protected    virtual    void    OnProtected(EventArgs    e);
[C++]    protected:    virtual    void    OnProtected(EventArgs*    e);
[VB]    Overridable Protected Sub OnProtected(ByVal e As EventArgs)
[JScript]    protected    function    OnProtected(e    :    EventArgs);
```

Description

Raises the **System.Windows.Forms.RichTextBox.Protected** event.

Raising an event invokes the event handler through a delegate. For more information, see . An **System.EventArgs** that contains the event data.

ggggggg)OnRightToLeftChanged

```
[C#]    protected    override    void    OnRightToLeftChanged(EventArgs    e);
[C++]    protected:    void    OnRightToLeftChanged(EventArgs*    e);
[VB]    Overrides Protected Sub OnRightToLeftChanged(ByVal e As EventArgs)
[JScript]    protected override function OnRightToLeftChanged(e : EventArgs);
```

hhhhhhh)OnSelectionChanged

```
[C#]    protected    virtual    void    OnSelectionChanged(EventArgs    e);
[C++]    protected:    virtual    void    OnSelectionChanged(EventArgs*    e);
[VB]    Overridable Protected Sub OnSelectionChanged(ByVal e As EventArgs)
[JScript]    protected    function    OnSelectionChanged(e    :    EventArgs);
```

Description

Raises the **System.Windows.Forms.RichTextBox.SelectionChanged** event.

Raising an event invokes the event handler through a delegate. For more information, see . An **System.EventArgs** that contains the event data.

iiiiiii) OnStyleChanged

```
[C#]    protected    override    void    OnStyleChanged(EventArgs    e);
[C++]    protected:    void    OnStyleChanged(EventArgs*    e);
[VB]    Overrides Protected Sub OnStyleChanged(ByVal e As EventArgs)
[JScript] protected override function OnStyleChanged(e : EventArgs);
```

Description

jjjjjjj) OnTextChanged

```
[C#]    protected    override    void    OnTextChanged(EventArgs    e);
[C++]    protected:    void    OnTextChanged(EventArgs*    e);
[VB]    Overrides Protected Sub OnTextChanged(ByVal e As EventArgs)
[JScript] protected override function OnTextChanged(e : EventArgs);
```

Description

Fires the event indicating that the text property has been changed. Inheriting controls should use this in favour of actually listening to the event, but should not forget to call `base.OnTextChanged()` to ensure that the event is still fired for external listeners. Event to send

kkkkkkkk)OnVScroll

```
[C#]    protected    virtual    void    OnVScroll(EventArgs    e);
[C++]    protected:    virtual    void    OnVScroll(EventArgs*    e);
```



```

1 [VB] Overridable Protected Sub OnVScroll(ByVal e As EventArgs)
2 [JScript] protected function OnVScroll(e : EventArgs);

```

Description

Raises the **System.Windows.Forms.RichTextBox.VScroll** event.

Raising an event invokes the event handler through a delegate. For more information, see . An **System.EventArgs** that contains the event data.

||||||| Paste

```

9 [C#] public void Paste(DataFormats.Format clipFormat);
10 [C++] public: void Paste(DataFormats.Format* clipFormat);
11 [VB] Public Sub Paste(ByVal clipFormat As DataFormats.Format)
12 [JScript] public function Paste(clipFormat : DataFormats.Format); Pastes the
13 contents of the Clipboard into the control.
14

```

Description

Pastes the contents of the Clipboard in the specified Clipboard format.

You can use this method to paste data from the clipboard into the control. This version of the

System.Windows.Forms.RichTextBox.Paste(System.Windows.Forms.DataFormats.Format) method is different from the

System.Windows.Forms.TextBoxBase.Paste method as it allows you to paste only text in a specified Clipboard format. You can use the

System.Windows.Forms.RichTextBox.CanPaste(System.Windows.Forms.DataFormats.Format) method to determine whether the data within the Clipboard is in the specified Clipboard format. You can then call this version of the

System.Windows.Forms.RichTextBox.Paste(System.Windows.Forms.DataFormats.Format) method to ensure that the paste operation is made with the appropriate data format. The Clipboard format in which the data should be obtained from the Clipboard.

mmmmmmmm)Redo

[C#]	public	void	Redo();
[C++]	public:	void	Redo();
[VB]	Public	Sub	Redo()
[JScript]	public	function	Redo();

Description

Reapplies the last operation that was undone in the control.

You can then use the **System.Windows.Forms.RichTextBox.Redo** method to reapply the last undo operation to the control. The **System.Windows.Forms.RichTextBox.CanRedo** method enables you to determine whether the last operation the user has undone can be reapplied to the control.

nnnnnnnn)SaveFile

[C#]	public	void	SaveFile(string	path);
[C++]	public:	void	SaveFile(String*	path);
[VB]	Public	Sub	SaveFile(ByVal	path As String)
[JScript]	public	function	SaveFile(path : String);	Saves the contents of the
System.Windows.Forms.RichTextBox			to	a file.

Description

Saves the contents of the **System.Windows.Forms.RichTextBox** to a Rich Text Format (RTF) file.

The **System.Windows.Forms.RichTextBox.SaveFile(System.String)** method enables you to save the entire contents of the control to an RTF file that can be used by other programs such as Microsoft Word and Windows WordPad. If the file name that is passed to the *path* parameter already exists at the

specified directory, the file will be overwritten without notice. You can use the **System.Windows.Forms.RichTextBox.LoadFile(System.String)** method to load the contents of a file into the **System.Windows.Forms.RichTextBox**. The name and location of the file to save.

oooooooo)SaveFile

[C#] public void SaveFile(Stream data, RichTextBoxStreamType fileType);

[C++] public: void SaveFile(Stream* data, RichTextBoxStreamType fileType);

[VB] Public Sub SaveFile(ByVal data As Stream, ByVal fileType As RichTextBoxStreamType)

[JScript] public function SaveFile(data : Stream, fileType : RichTextBoxStreamType);

Description

Saves the contents of a **System.Windows.Forms.RichTextBox** control to an open data stream.

This version of the **System.Windows.Forms.RichTextBox.SaveFile(System.String)** method enables you to save the entire contents of the control to the data stream that is already opened. The data stream can then save the information to a file. You can use the **System.Windows.Forms.RichTextBox.LoadFile(System.String)** method to load the contents of a file into the **System.Windows.Forms.RichTextBox**. The data stream that contains the file to save to. One of the **System.Windows.Forms.RichTextBoxStreamType** values.

pppppppp)SaveFile

[C#] public void SaveFile(string path, RichTextBoxStreamType fileType);

[C++] public: void SaveFile(String* path, RichTextBoxStreamType fileType);

[VB] Public Sub SaveFile(ByVal path As String, ByVal fileType As

RichTextBoxStreamType)

[JScript] public function SaveFile(path : String, fileType :
RichTextBoxStreamType);

Description

Saves the contents of the **System.Windows.Forms.RichTextBox** to a specific type of file.

The **System.Windows.Forms.RichTextBox.SaveFile(System.String)** method enables you to save the entire contents of the control to an RTF file that can be used by other programs such as Microsoft Word and Windows WordPad. If the file name that is passed to the *path* parameter already exists at the specified directory, the file will be overwritten without notice. You can use the **System.Windows.Forms.RichTextBox.LoadFile(System.String)** method to load the contents of a file into the **System.Windows.Forms.RichTextBox**. The name and location of the file to save. One of the **System.Windows.Forms.RichTextBoxStreamType** values.

qqqqqqqq)WndProc

[C#] protected override void WndProc(ref Message m);

[C++] protected: void WndProc(Message* m);

[VB] Overrides Protected Sub WndProc(ByRef m As Message)

[JScript] protected override function WndProc(m : Message);

RichTextBoxFinds enumeration (System.Windows.Forms)

a) WndProc

Description

Specifies how a text search is carried out in a **System.Windows.Forms.RichTextBox** control.

An application locates text in the **System.Windows.Forms.RichTextBox** control by calling the **System.Windows.Forms.RichTextBox.Find(System.String)** method of the **System.Windows.Forms.RichTextBox** control. This enumeration enables you to specify how the search is performed when the **System.Windows.Forms.RichTextBox.Find(System.String)** method is called. You can combine one or more values from this enumeration to specify more than one search option when calling the **System.Windows.Forms.RichTextBox.Find(System.String)** method.

b) WndProc

[C#]	public	const	RichTextBoxFinds	MatchCase;
[C++]	public:	const	RichTextBoxFinds	MatchCase;
[VB]	Public	Const	MatchCase As	RichTextBoxFinds
[JScript]	public	var	MatchCase :	RichTextBoxFinds;

Description

Locate only instances of the search text that have the exact casing.

c) WndProc

[C#]	public	const	RichTextBoxFinds	NoHighlight;
[C++]	public:	const	RichTextBoxFinds	NoHighlight;
[VB]	Public	Const	NoHighlight As	RichTextBoxFinds
[JScript]	public	var	NoHighlight :	RichTextBoxFinds;

Description

The search text, if found, should not be highlighted.

d) *WndProc*

[C#]	public	const	RichTextBoxFinds	None;
[C++]	public:	const	RichTextBoxFinds	None;
[VB]	Public	Const	None	As RichTextBoxFinds
[JScript]	public	var	None	: RichTextBoxFinds;

Description

Locate all instances of the search text, whether the instances found in the search are whole words or not.

e) *WndProc*

[C#]	public	const	RichTextBoxFinds	Reverse;
[C++]	public:	const	RichTextBoxFinds	Reverse;
[VB]	Public	Const	Reverse	As RichTextBoxFinds
[JScript]	public	var	Reverse	: RichTextBoxFinds;

Description

The search starts at the end of the control's document and searches to the beginning of the document.

f) *WndProc*

[C#]	public	const	RichTextBoxFinds	WholeWord;
[C++]	public:	const	RichTextBoxFinds	WholeWord;
[VB]	Public	Const	WholeWord	As RichTextBoxFinds
[JScript]	public	var	WholeWord	: RichTextBoxFinds;

Description

Locate only instances of the search text that are whole words.

RichTextBoxScrollBars enumeration (System.Windows.Forms)

a) ToString

Description

Specifies the type of scroll bars to display in a **System.Windows.Forms.RichTextBox** control.

Use the members of this enumeration to set the value of the **System.Windows.Forms.RichTextBox.ScrollBars** property of the **System.Windows.Forms.RichTextBox** control.

b) ToString

[C#]	public	const	RichTextBoxScrollBars	Both;
[C++]	public:	const	RichTextBoxScrollBars	Both;
[VB]	Public	Const	Both As	RichTextBoxScrollBars
[JScript]	public	var	Both :	RichTextBoxScrollBars;

Description

Display both a horizontal and a vertical scroll bar when needed.

c) ToString

[C#]	public	const	RichTextBoxScrollBars	ForcedBoth;
[C++]	public:	const	RichTextBoxScrollBars	ForcedBoth;

```

1  [VB]      Public      Const      ForcedBoth      As      RichTextBoxScrollBars
2  [JScript] public      var      ForcedBoth      :      RichTextBoxScrollBars;

```

Description

Always display both a horizontal and a vertical scroll bar.

d) ToString

```

8  [C#]      public      const      RichTextBoxScrollBars      ForcedHorizontal;
9  [C++]     public:     const      RichTextBoxScrollBars      ForcedHorizontal;
10 [VB]      Public      Const      ForcedHorizontal      As      RichTextBoxScrollBars
11 [JScript] public      var      ForcedHorizontal      :      RichTextBoxScrollBars;

```

Description

Always display a horizontal scroll bar.

e) ToString

```

17 [C#]      public      const      RichTextBoxScrollBars      ForcedVertical;
18 [C++]     public:     const      RichTextBoxScrollBars      ForcedVertical;
19 [VB]      Public      Const      ForcedVertical      As      RichTextBoxScrollBars
20 [JScript] public      var      ForcedVertical      :      RichTextBoxScrollBars;

```

Description

Always display a vertical scroll bar.

f) ToString

[C#]	public	const	RichTextBoxScrollBars	Horizontal;
[C++]	public:	const	RichTextBoxScrollBars	Horizontal;
[VB]	Public	Const	Horizontal As	RichTextBoxScrollBars
[JScript]	public	var	Horizontal :	RichTextBoxScrollBars;

Description

Display a horizontal scroll bar only when text is longer than the width of the control.

g) ToString

[C#]	public	const	RichTextBoxScrollBars	None;
[C++]	public:	const	RichTextBoxScrollBars	None;
[VB]	Public	Const	None As	RichTextBoxScrollBars
[JScript]	public	var	None :	RichTextBoxScrollBars;

Description

No scroll bars are displayed.

h) ToString

[C#]	public	const	RichTextBoxScrollBars	Vertical;
[C++]	public:	const	RichTextBoxScrollBars	Vertical;
[VB]	Public	Const	Vertical As	RichTextBoxScrollBars
[JScript]	public	var	Vertical :	RichTextBoxScrollBars;

Description

Display a vertical scroll bar only when text is longer than the height of the control.

RichTextBoxSelectionAttribute enumeration (System.Windows.Forms)

a) ToString

Description

Specifies whether any characters in the current selection have the style or attribute.

The **System.Windows.Forms.RichTextBox** class uses this enumeration internally.

b) ToString

[C#]	public	const	RichTextBoxSelectionAttribute	All;
[C++]	public:	const	RichTextBoxSelectionAttribute	All;
[VB]	Public	Const	All As	RichTextBoxSelectionAttribute
[JScript]	public	var	All :	RichTextBoxSelectionAttribute;

Description

All characters.

c) ToString

[C#]	public	const	RichTextBoxSelectionAttribute	Mixed;
[C++]	public:	const	RichTextBoxSelectionAttribute	Mixed;

```

1  [VB]    Public    Const    Mixed    As    RichTextBoxSelectionAttribute
2  [JScript]    public    var    Mixed    :    RichTextBoxSelectionAttribute;

```

Description

Some but not all characters.

d) ToString

```

8  [C#]    public    const    RichTextBoxSelectionAttribute    None;
9  [C++]    public:    const    RichTextBoxSelectionAttribute    None;
10 [VB]    Public    Const    None    As    RichTextBoxSelectionAttribute
11 [JScript]    public    var    None    :    RichTextBoxSelectionAttribute;

```

Description

No characters.

RichTextBoxSelectionTypes enumeration (System.Windows.Forms)

a) ToString

Description

Specifies the type of selection in a **System.Windows.Forms.RichTextBox** control.

Use the members of this enumeration to determine the value of the **System.Windows.Forms.RichTextBox.SelectionType** property of the **System.Windows.Forms.RichTextBox** class. The **System.Windows.Forms.RichTextBox.SelectionType** property can return any combination of values from this enumeration.

b) ToString

```
[C#]      public      const      RichTextBoxSelectionTypes      Empty;  
[C++]     public:     const      RichTextBoxSelectionTypes      Empty;  
[VB]      Public      Const      Empty      As      RichTextBoxSelectionTypes  
[JScript] public      var      Empty      :      RichTextBoxSelectionTypes;
```

Description

No text is selected in the current selection.

c) ToString

```
[C#]      public      const      RichTextBoxSelectionTypes      MultiChar;  
[C++]     public:     const      RichTextBoxSelectionTypes      MultiChar;  
[VB]      Public      Const      MultiChar      As      RichTextBoxSelectionTypes  
[JScript] public      var      MultiChar      :      RichTextBoxSelectionTypes;
```

Description

More than one character is selected.

d) ToString

```
[C#]      public      const      RichTextBoxSelectionTypes      MultiObject;  
[C++]     public:     const      RichTextBoxSelectionTypes      MultiObject;  
[VB]      Public      Const      MultiObject      As      RichTextBoxSelectionTypes  
[JScript] public      var      MultiObject      :      RichTextBoxSelectionTypes;
```

Description

More than one Object Linking and Embedding (OLE) object is selected.

e) ToString

[C#] public const RichTextBoxSelectionTypes Object;

[C++] public: const RichTextBoxSelectionTypes Object;

[VB] Public Const Object As RichTextBoxSelectionTypes

[JScript] public var Object : RichTextBoxSelectionTypes;

Description

At least one Object Linking and Embedding (OLE) object is selected.

f) ToString

[C#] public const RichTextBoxSelectionTypes Text;

[C++] public: const RichTextBoxSelectionTypes Text;

[VB] Public Const Text As RichTextBoxSelectionTypes

[JScript] public var Text : RichTextBoxSelectionTypes;

Description

The current selection contains only text.

RichTextBoxStreamType enumeration (System.Windows.Forms)

a) *ToString*

Description

Specifies the types of input and output streams used to load and save data in the **System.Windows.Forms.RichTextBox** control.

Use the members of this enumeration when calling the **System.Windows.Forms.RichTextBox.LoadFile(System.String)** and **System.Windows.Forms.RichTextBox.SaveFile(System.String)** methods of the **System.Windows.Forms.RichTextBox** control.

b) *ToString*

[C#] public const RichTextBoxStreamType PlainText;

[C++] public: const RichTextBoxStreamType PlainText;

[VB] Public Const PlainText As RichTextBoxStreamType

[JScript] public var PlainText : RichTextBoxStreamType;

Description

A plain text stream that includes spaces in places of Object Linking and Embedding (OLE) objects.

c) *ToString*

[C#] public const RichTextBoxStreamType RichNoOleObjs;

[C++] public: const RichTextBoxStreamType RichNoOleObjs;

[VB] Public Const RichNoOleObjs As RichTextBoxStreamType

[JScript] public var RichNoOleObjs : RichTextBoxStreamType;

1
2 *Description*

3 A Rich Text Format (RTF) stream with spaces in place of OLE objects. This value
4 is only valid for use with the
5 **System.Windows.Forms.RichTextBox.SaveFile(System.String)** method of
6 the **System.Windows.Forms.RichTextBox** control.

7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
d) ToString

[C#] public const RichTextBoxStreamType RichText;
[C++] public: const RichTextBoxStreamType RichText;
[VB] Public Const RichText As RichTextBoxStreamType
[JScript] public var RichText : RichTextBoxStreamType;

Description

A Rich Text Format (RTF) stream.

e) ToString

[C#] public const RichTextBoxStreamType TextTextOleObjs;
[C++] public: const RichTextBoxStreamType TextTextOleObjs;
[VB] Public Const TextTextOleObjs As RichTextBoxStreamType
[JScript] public var TextTextOleObjs : RichTextBoxStreamType;

Description

A plain text stream with a textual representation of OLE objects. This value is
only valid for use with the
System.Windows.Forms.RichTextBox.SaveFile(System.String) method of
the **System.Windows.Forms.RichTextBox** control.

f) ToString

```
[C#]    public    const    RichTextBoxStreamType    UnicodePlainText;
[C++]   public:   const    RichTextBoxStreamType    UnicodePlainText;
[VB]    Public    Const    UnicodePlainText    As    RichTextBoxStreamType
[JScript] public  var    UnicodePlainText    :    RichTextBoxStreamType;
```

Description

A text stream that contains spaces in place of Object Linking and Embedding (OLE) objects. The text is encoded in Unicode.

RichTextBoxWordPunctuations enumeration (System.Windows.Forms)

a) ToString

Description

Specifies the types of punctuation tables that can be used with the **System.Windows.Forms.RichTextBox** control's word-wrapping and word-breaking features.

b) ToString

```
[C#]    public    const    RichTextBoxWordPunctuations    All;
[C++]   public:   const    RichTextBoxWordPunctuations    All;
[VB]    Public    Const    All    As    RichTextBoxWordPunctuations
[JScript] public  var    All    :    RichTextBoxWordPunctuations;
```

Description

Used as a mask.

c) *ToString*

```
[C#]      public      const      RichTextBoxWordPunctuations      Custom;  
[C++]     public:     const      RichTextBoxWordPunctuations      Custom;  
[VB]      Public      Const      Custom      As      RichTextBoxWordPunctuations  
[JScript] public      var      Custom      :      RichTextBoxWordPunctuations;
```

Description

Use a custom defined punctuation table.

d) *ToString*

```
[C#]      public      const      RichTextBoxWordPunctuations      Level1;  
[C++]     public:     const      RichTextBoxWordPunctuations      Level1;  
[VB]      Public      Const      Level1      As      RichTextBoxWordPunctuations  
[JScript] public      var      Level1      :      RichTextBoxWordPunctuations;
```

Description

Use pre-defined Level 1 punctuation table as default.

e) *ToString*

```
[C#]      public      const      RichTextBoxWordPunctuations      Level2;  
[C++]     public:     const      RichTextBoxWordPunctuations      Level2;  
[VB]      Public      Const      Level2      As      RichTextBoxWordPunctuations  
[JScript] public      var      Level2      :      RichTextBoxWordPunctuations;
```

1
2 *Description*

3 Use pre-defined Level 2 punctuation table as default.

4 RightToLeft enumeration (System.Windows.Forms)

5 *a) ToString*

6
7
8 *Description*

9 Specifies a value indicating whether the text appears from right to left, such as
10 when using Hebrew or Arabic fonts.

11 This enumeration is used by **System.Windows.Forms.ContextMenu** ,
12 **System.Windows.Forms.MainMenu** ,
13 **System.Windows.Forms.ProgressBar** ,
14 **System.Text.RegularExpressions.Regex** and
15 **System.Windows.Forms.Control** .

16
17 *b) ToString*

18 [C#]	public	const	RightToLeft	Inherit;
19 [C++]	public:	const	RightToLeft	Inherit;
20 [VB]	Public	Const	Inherit As	RightToLeft
21 [JScript]	public	var	Inherit :	RightToLeft;

22
23 *Description*

24 The direction the text read is inherited from the parent control.

25 *c) ToString*

[C#]	public	const	RightToLeft	No;
------	--------	-------	-------------	-----

[C++]	public:	const		RightToLeft	No;
[VB]	Public	Const	No	As	RightToLeft
[JScript]	public	var	No	:	RightToLeft;

Description

The text reads from left to right. This is the default.

d) ToString

[C#]	public	const		RightToLeft	Yes;
[C++]	public:	const		RightToLeft	Yes;
[VB]	Public	Const	Yes	As	RightToLeft
[JScript]	public	var	Yes	:	RightToLeft;

Description

The text reads from right to left.

SaveFileDialog class (System.Windows.Forms)

a) ToString

Description

Represents a common dialog box that allows the user to specify options for saving a file. This class cannot be inherited.

This class allows you to open and overwrite an existing file or create a new file.

b) SaveFileDialog

Example Syntax:

c) *ToString*

[C#] public SaveFileDialog();
[C++] public: SaveFileDialog();
[VB] Public Sub New()
[JScript] public function SaveFileDialog();

d) *AddExtension*

e) *CheckFileExists*

f) *CheckPathExists*

g) *Container*

h) *CreatePrompt*

i) *ToString*

Description

Gets or sets a value indicating whether the dialog box prompts the user for permission to create a file if the user specifies a file that does not exist.

- j) *DefaultExt*
- k) *DereferenceLinks*
- l) *DesignMode*
- m) *Events*
- n) *FileName*
- o) *FileNames*
- p) *Filter*
- q) *FilterIndex*
- r) *InitialDirectory*
- s) *Instance*
- t) *Options*
- u) *OverwritePrompt*
- v) *ToString*

Description

Gets or sets a value indicating whether the Save As dialog box displays a warning if the user specifies a file name that already exists.

w) *RestoreDirectory*

x) *ShowHelp*

y) *Site*

z) *Title*

aa) *ValidateNames*

bb) *OpenFile*

[C#] public Stream OpenFile();

[C++] public: Stream* OpenFile();

[VB] Public Function OpenFile() As Stream

[JScript] public function OpenFile() : Stream;

Description

Opens the file with read/write permission selected by the user.

Return Value: The read/write file selected by the user.

For security purposes, this method creates a new file with the selected name and opens it with read/write permissions. This can cause unintentional loss of data if you select an existing file to save to. To save data to an existing file while retaining existing data, use the **System.IO.File** class to open the file using the file name returned in the **System.Windows.Forms.FileDialog.FileName** property.

cc) *Reset*

[C#] public override void Reset();

[C++] public: void Reset();

[VB] Overrides Public Sub Reset()

[JScript] public override function Reset();

1
2 *Description*

3 Resets all dialog box options to their default values.

4 Screen class (System.Windows.Forms)

5 a) *ToString*

6
7
8 *Description*

9 Represents a display device or multiple display devices on a single system.

10 The constructor for this object is public, so you cannot explicitly create a
11 **System.Windows.Forms.Screen** object. The object is created when you call
its public methods.

12 b) *AllScreens*

13 c) *ToString*

14
15 [C#] public static Screen[] AllScreens {get;}

16 [C++] public: __property static Screen* get_AllScreens();

17 [VB] Public Shared ReadOnly Property AllScreens As Screen ()

18 [JScript] public static function get AllScreens() : Screen[];

19
20 *Description*

21 Gets an array of all displays on the system.
22
23
24
25

d) *Bounds*

e) *ToString*

[C#] public Rectangle Bounds {get;}

[C++] public: __property Rectangle get_Bounds();

[VB] Public ReadOnly Property Bounds As Rectangle

[JScript] public function get Bounds() : Rectangle;

Description

Gets the bounds of the display.

f) *DeviceName*

g) *ToString*

[C#] public string DeviceName {get;}

[C++] public: __property String* get_DeviceName();

[VB] Public ReadOnly Property DeviceName As String

[JScript] public function get DeviceName() : String;

Description

Gets the device name associated with a display.

h) *Primary*

i) *ToString*

[C#] public bool Primary {get;}

[C++] public: __property bool get_Primary();


```

1 [VB]      Public      ReadOnly      Property      Primary      As      Boolean
2 [JScript]      public      function      get      Primary()      :      Boolean;

```

Description

Gets a value indicating whether a particular display is the primary device.

j) PrimaryScreen

k) ToString

```

9 [C#]      public      static      Screen      PrimaryScreen      {get;}
10 [C++]      public:      __property      static      Screen*      get_PrimaryScreen();
11 [VB]      Public      Shared      ReadOnly      Property      PrimaryScreen      As      Screen
12 [JScript]      public      static      function      get      PrimaryScreen()      :      Screen;

```

Description

Gets the primary display.

For a single display system, the primary display is the only display.

l) WorkingArea

m) ToString

```

20 [C#]      public      Rectangle      WorkingArea      {get;}
21 [C++]      public:      __property      Rectangle      get_WorkingArea();
22 [VB]      Public      ReadOnly      Property      WorkingArea      As      Rectangle
23 [JScript]      public      function      get      WorkingArea()      :      Rectangle;

```

1
2 *Description*

3 Gets the working area of the display. The working area is the desktop area of the
4 display, excluding taskbars, docked windows, and docked tool bars.

5 The working area is the desktop area of the display, excluding taskbars, docked
6 windows, and docked tool bars.

7
8 *n) Equals*

9 [C#] public override bool Equals(object obj);

10 [C++] public: bool Equals(Object* obj);

11 [VB] Overrides Public Function Equals(ByVal obj As Object) As Boolean

12 [JScript] public override function Equals(obj : Object) : Boolean;

13 *Description*

14 Gets or sets a value indicating whether the specified object is equal to this
15 **Screen** .

16 *Return Value:* **true** if the specified object is equal to this
17 **System.Windows.Forms.Screen** ; otherwise, **false** . The object to compare
18 to this **System.Windows.Forms.Screen** .

19 *o) FromControl*

20 [C#] public static Screen FromControl(Control control);

21 [C++] public: static Screen* FromControl(Control* control);

22 [VB] Public Shared Function FromControl(ByVal control As Control) As Screen

23 [JScript] public static function FromControl(control : Control) : Screen;

24 *Description*

Retrieves a **System.Windows.Forms.Screen** for the display that contains the largest portion of the specified control.

Return Value: A **System.Windows.Forms.Screen** for the display that contains the largest region of the specified control. In multiple display environments where no display contains the control, the display closest to the specified control is returned. A **System.Windows.Forms.Control** for which to retrieve a **System.Windows.Forms.Screen**.

p) *FromHandle*

[C#] public static Screen FromHandle(IntPtr hwnd);

[C++] public: static Screen* FromHandle(IntPtr hwnd);

[VB] Public Shared Function FromHandle(ByVal hwnd As IntPtr) As Screen

[JScript] public static function FromHandle(hwnd : IntPtr) : Screen;

Description

Retrieves a **System.Windows.Forms.Screen** for the display that contains the largest portion of the of the object referred to by the specified handle.

Return Value: A **System.Windows.Forms.Screen** for the display that contains the largest region of the object. In multiple display environments where no display contains any portion of the specified window, the display closest to the object is returned. The window handle for which to retrieve the **System.Windows.Forms.Screen**.

q) *FromPoint*

[C#] public static Screen FromPoint(Point point);

[C++] public: static Screen* FromPoint(Point point);

[VB] Public Shared Function FromPoint(ByVal point As Point) As Screen

[JScript] public static function FromPoint(point : Point) : Screen;

Description

Retrieves a **System.Windows.Forms.Screen** for the display that contains the specified point.

Return Value: A **System.Windows.Forms.Screen** for the display that contains the point. In multiple display environments where no display contains the point, the display closest to the specified point is returned. A **System.Drawing.Point** that specifies the location for which to retrieve a

System.Windows.Forms.Screen.

r) FromRectangle

[C#] public static Screen FromRectangle(Rectangle rect);

[C++] public: static Screen* FromRectangle(Rectangle rect);

[VB] Public Shared Function FromRectangle(ByVal rect As Rectangle) As Screen

[JScript] public static function FromRectangle(rect : Rectangle) : Screen;

Description

Retrieves a **System.Windows.Forms.Screen** for the display that contains the largest portion of the rectangle.

Return Value: A **System.Windows.Forms.Screen** for the display that contains the largest region of the specified rectangle. In multiple display environments where no display contains the rectangle, the display closest to the rectangle is returned. A **System.Drawing.Rectangle** that specifies the area for which to retrieve the display.

s) GetBounds

[C#] public static Rectangle GetBounds(Control ctl);

[C++] public: static Rectangle GetBounds(Control* ctl);

[VB] Public Shared Function GetBounds(ByVal ctl As Control) As Rectangle

[JScript] public static function GetBounds(ctl : Control) : Rectangle;

Description

Retrieves the bounds of the display that contains the largest portion of the specified control.

Return Value: A **System.Drawing.Rectangle** that specifies the bounds of the display that contains the specified control. In multiple display environments where no display contains the specified control, the display closest to the control is returned. The **System.Windows.Forms.Control** for which to retrieve the display bounds.

t) GetBounds

[C#] public static Rectangle GetBounds(Point pt);

[C++] public: static Rectangle GetBounds(Point pt);

[VB] Public Shared Function GetBounds(ByVal pt As Point) As Rectangle

[JScript] public static function GetBounds(pt : Point) : Rectangle; Retrieves the

bounds of the display.

Description

Retrieves the bounds of the display that contains the specified point.

Return Value: A **System.Drawing.Rectangle** that specifies the bounds of the display that contains the specified point. In multiple display environments where no display contains the specified point, the display closest to the point is returned. A **System.Drawing.Point** that specifies the coordinates for which to retrieve the display bounds.

u) GetBounds

[C#] public static Rectangle GetBounds(Rectangle rect);

[C++] public: static Rectangle GetBounds(Rectangle rect);

[VB] Public Shared Function GetBounds(ByVal rect As Rectangle) As Rectangle

[JScript] public static function GetBounds(rect : Rectangle) : Rectangle;

Description

Retrieves the bounds of the display that contains the largest portion of the specified rectangle.

Return Value: A **System.Drawing.Rectangle** that specifies the bounds of the display that contains the specified rectangle. In multiple display environments where no monitor contains the specified rectangle, the monitor closest to the rectangle is returned. A **System.Drawing.Rectangle** that specifies the area for which to retrieve the display bounds.

v) *GetHashCode*

[C#] public override int GetHashCode();

[C++] public: int GetHashCode();

[VB] Overrides Public Function GetHashCode() As Integer

[JScript] public override function GetHashCode() : int;

Description

Computes and retrieves a hash code for an object.

Return Value: A hash code for an object.

w) *GetWorkingArea*

[C#] public static Rectangle GetWorkingArea(Control ctl);

[C++] public: static Rectangle GetWorkingArea(Control* ctl);

[VB] Public Shared Function GetWorkingArea(ByVal ctl As Control) As Rectangle

[JScript] public static function GetWorkingArea(ctl : Control) : Rectangle;

Description

Retrieves the working area for the display that contains the largest region of the specified control. The working area is the desktop area of the display, excluding taskbars, docked windows, and docked tool bars.

Return Value: A **System.Drawing.Rectangle** that specifies the working area. In multiple display environments where no display contains the specified control, the display closest to the control is returned. The **System.Windows.Forms.Control** for which to retrieve the working area.

x) *GetWorkingArea*

[C#] public static Rectangle GetWorkingArea(Point pt);

[C++] public: static Rectangle GetWorkingArea(Point pt);

[VB] Public Shared Function GetWorkingArea(ByVal pt As Point) As Rectangle

[JScript] public static function GetWorkingArea(pt : Point) : Rectangle; Retrieves

the working area of a display.

Description

Retrieves the working area closest to the specified point. The working area is the desktop area of the display, excluding taskbars, docked windows, and docked tool bars.

Return Value: A **System.Drawing.Rectangle** that specifies the working area. In multiple display environments where no display contains the specified point, the display closest to the point is returned. A **System.Drawing.Point** that specifies the coordinates for which to retrieve the working area.

y) *GetWorkingArea*

[C#] public static Rectangle GetWorkingArea(Rectangle rect);

[C++] public: static Rectangle GetWorkingArea(Rectangle rect);

[VB] Public Shared Function GetWorkingArea(ByVal rect As Rectangle) As Rectangle

[JScript] public static function GetWorkingArea(rect : Rectangle) : Rectangle;

Description

Retrieves the working area for the display that contains the largest portion of the specified rectangle. The working area is the desktop area of the display, excluding taskbars, docked windows, and docked tool bars.

Return Value: A **System.Drawing.Rectangle** that specifies the working area. In multiple display environments where no display contains the specified rectangle, the display closest to the rectangle is returned. The **System.Drawing.Rectangle** that specifies the area for which to retrieve the working area.

z) *ToString*

[C#]	public	override	string	ToString();
[C++]	public:		String*	ToString();
[VB]	Overrides	Public	Function	ToString() As String
[JScript]	public	override	function	ToString() : String;

Description

Retrieves a string representing this object.

Return Value: A string representation of the object.

ScrollableControl class (System.Windows.Forms)

a) *ToString*

Description

Defines a base class for controls that support auto-scrolling behavior.

You do not typically use this class directly. The **System.Windows.Forms.ContainerControl** and **System.Windows.Forms.Panel** classes inherit from this class.

b) ToString

[C#] protected const int ScrollStateAutoScrolling;
[C++] protected: const int ScrollStateAutoScrolling;
[VB] Protected Const ScrollStateAutoScrolling As Integer
[JScript] protected var ScrollStateAutoScrolling : int;

Description

c) ToString

[C#] protected const int ScrollStateFullDrag;
[C++] protected: const int ScrollStateFullDrag;
[VB] Protected Const ScrollStateFullDrag As Integer
[JScript] protected var ScrollStateFullDrag : int;

Description

d) ToString

[C#] protected const int ScrollStateHScrollVisible;
[C++] protected: const int ScrollStateHScrollVisible;
[VB] Protected Const ScrollStateHScrollVisible As Integer
[JScript] protected var ScrollStateHScrollVisible : int;

Description

e) ToString

```
[C#]      protected      const      int      ScrollStateUserHasScrolled;
[C++]      protected:      const      int      ScrollStateUserHasScrolled;
[VB]      Protected      Const      ScrollStateUserHasScrolled      As      Integer
[JScript]      protected      var      ScrollStateUserHasScrolled      :      int;
```

Description

f) ToString

```
[C#]      protected      const      int      ScrollStateVScrollVisible;
[C++]      protected:      const      int      ScrollStateVScrollVisible;
[VB]      Protected      Const      ScrollStateVScrollVisible      As      Integer
[JScript]      protected      var      ScrollStateVScrollVisible      :      int;
```

Description

g) ScrollableControl

Example Syntax:

h) ToString

```
[C#]      public      ScrollableControl();
[C++]      public:      ScrollableControl();
[VB]      Public      Sub      New()
[JScript]      public      function      ScrollableControl();
```

Description

Initializes a new instance of the **System.Windows.Forms.ScrollableControl** class.

- i) *AccessibilityObject*
- j) *AccessibleDefaultActionDescription*
- k) *AccessibleDescription*
- l) *AccessibleName*
- m) *AccessibleRole*
- n) *AllowDrop*
- o) *Anchor*
- p) *AutoScroll*
- q) *ToString*

Description

Gets or sets a value indicating whether the container will allow the user to scroll to any controls placed outside of its visible boundaries.

When **true** , this property allows the container to have a virtual size that is larger than its visible boundaries.

- r) *AutoScrollMargin*
- s) *ToString*

[C#] public Size AutoScrollMargin {get; set;}

[C++] public: __property Size get_AutoScrollMargin();public: __property void

1 set_AutoScrollMargin(Size);

2 [VB] Public Property AutoScrollMargin As Size

3 [JScript] public function get AutoScrollMargin() : Size;public function set

4 AutoScrollMargin(Size);

5
6 *Description*

7 Gets or sets the size of the auto-scroll margin.

8 The auto-scroll-margin is the distance between any child controls and the edges
9 of the scrollable parent control. The

10 **System.Windows.Forms.ScrollableControl.AutoScrollMargin** size is added
11 to the size of any child controls contained in the scrollable control to determine
12 whether or not scroll bars are needed. The

13 **System.Windows.Forms.ScrollableControl.AutoScrollMargin** property is
14 evaluated when the parent scrollable control is resized or the individual child
15 controls are brought into view, and is used to determine if scroll bars need to be
16 displayed. Docked controls are excluded from the calculations that determine if
17 scroll bars need to be displayed.

14 t) *AutoScrollMinSize*

15 u) *ToString*

17 [C#] public Size AutoScrollMinSize {get; set;}

18 [C++] public: __property Size get_AutoScrollMinSize();public: __property void

19 set_AutoScrollMinSize(Size);

20 [VB] Public Property AutoScrollMinSize As Size

21 [JScript] public function get AutoScrollMinSize() : Size;public function set

22 AutoScrollMinSize(Size);

23
24 *Description*

25 Gets or sets the minimum size of the auto-scroll.

The **System.Windows.Forms.ScrollableControl.AutoScrollMinSize** property is used to manage the screen size allocated to the automatic scroll bars.

v) *AutoScrollPosition*

w) *ToString*

[C#] public Point AutoScrollPosition {get; set;}

[C++] public: __property Point get_AutoScrollPosition();public: __property void
set_AutoScrollPosition(Point);

[VB] Public Property AutoScrollPosition As Point

[JScript] public function get AutoScrollPosition() : Point;public function set
AutoScrollPosition(Point);

Description

Gets or sets the location of the auto-scroll position.

The **System.Windows.Forms.ScrollableControl.AutoScrollPosition** property is used to adjust the position of controls contained on the scrollable control.

- x) *BackColor*
- y) *BackgroundImage*
- z) *BindingContext*
- aa) *Bottom*
- bb) *Bounds*
- cc) *CanFocus*
- dd) *CanSelect*
- ee) *Capture*
- ff) *CausesValidation*
- gg) *ClientRectangle*
- hh) *ClientSize*
- ii) *CompanyName*
- jj) *Container*
- kk) *ContainsFocus*
- ll) *ContextMenu*
- mm) *Controls*
- nn) *Created*
- oo) *CreateParams*
- pp) *ToString*

Description

Retrieves the CreateParams used to create the window. If a subclass overrides this function, it must call the base implementation.

qq) Cursor
rr) DataBindings
ss) DefaultImeMode
tt) DefaultSize
uu) DesignMode
vv) DisplayRectangle
ww) ToString

Description

Retreives the current display rectangle. The display rectangle is the virtual display area that is used to layout components. The position and dimensions of the Form's display rectangle change during autoScroll.

xx) Disposing
yy) Dock
zz) DockPadding
aaa) ToString

Description

Gets the dock padding settings for all edges of the control.

This property controls the border inside of this control for docked components.

bbb) Enabled

ccc) Events

ddd) Focused

eee) Font

fff) FontHeight

ggg) ForeColor

hhh) Handle

iii) HasChildren

jjj) Height

kkk) HScroll

lll) ToString

Description

Gets or sets a value indicating whether the horizontal scroll bar is visible.

1 *mmm) ImeMode*

2 *nnn) InvokeRequired*

3 *ooo) IsAccessible*

4 *ppp) IsDisposed*

5 *qqq) IsHandleCreated*

6 *rrr) Left*

7 *sss) Location*

8 *ttt) Name*

9 *uuu) Parent*

10 *vvv) ProductName*

11 *www) ProductVersion*

12 *xxx) RecreatingHandle*

13 *yyy) Region*

14 *zzz) RenderRightToLeft*

15 *aaaa) ResizeRedraw*

16 *bbbb) Right*

17 *cccc) RightToLeft*

18 *dddd) ShowFocusCues*

19 *eeee) ShowKeyboardCues*

20 *ffff) Site*

21 *gggg) Size*

22 *hhhh) TabIndex*

23 *iiii) TabStop*

jjjj) Tag

kkkk) Text

III) Top

mmmm)TopLevelControl

nnnn) Visible

oooo) VScroll

pppp) *ToString*

Description

Gets or sets a value indicating whether the vertical scroll bar is visible.

qqqq) Width

rrrr) WindowTarget

ssss) AdjustFormScrollbars

```
[C#] protected virtual void AdjustFormScrollbars(bool displayScrollbars);
```

```
[C++] protected: virtual void AdjustFormScrollbars(bool displayScrollbars);
```

[VB] Overridable Protected Sub AdjustFormScrollbars(ByVal displayScrollbars

As Boolean)

```
[JScript] protected function AdjustFormScrollbars(displayScrollbars : Boolean);
```

Description

Adjusts the auto-scroll bars on the container based on the current control positions and the control currently selected. **true** to show the scroll bars; otherwise, **false**.

tttt) GetScrollState

```
1
2 [C#]      protected      bool      GetScrollState(int      bit);
3
4 [C++]      protected:      bool      GetScrollState(int      bit);
5
6 [VB] Protected Function GetScrollState(ByVal bit As Integer) As Boolean
7
8 [JScript] protected function GetScrollState(bit : int) : Boolean;
```

Description

Tests a given scroll state bit to determine if it is set.

Return Value: **true** if the given bit is set; otherwise, **false** . The scroll state bit to test.

uuuu) OnLayout

```
11
12
13 [C#]      protected      override      void      OnLayout(LayoutEventArgs      levent);
14
15 [C++]      protected:      void      OnLayout(LayoutEventArgs*      levent);
16
17 [VB] Overrides Protected Sub OnLayout(ByVal levent As LayoutEventArgs)
18
19 [JScript] protected override function OnLayout(levent : LayoutEventArgs);
```

Description

Forces the layout of any docked or anchored child controls.

vvvv) OnMouseWheel

```
20
21
22 [C#]      protected      override      void      OnMouseWheel(MouseEventArgs      e);
23
24 [C++]      protected:      void      OnMouseWheel(MouseEventArgs*      e);
25
26 [VB] Overrides Protected Sub OnMouseWheel(ByVal e As MouseEventArgs)
27
28 [JScript] protected override function OnMouseWheel(e : MouseEventArgs);
```

Description

Handles mouse wheel processing for our scrollbars.

www)OnVisibleChanged

[C#] protected override void OnVisibleChanged(EventArgs e);

[C++] protected: void OnVisibleChanged(EventArgs* e);

[VB] Overrides Protected Sub OnVisibleChanged(ByVal e As EventArgs)

[JScript] protected override function OnVisibleChanged(e : EventArgs);

Description

xxxx) ScaleCore

[C#] protected override void ScaleCore(float dx, float dy);

[C++] protected: void ScaleCore(float dx, float dy);

[VB] Overrides Protected Sub ScaleCore(ByVal dx As Single, ByVal dy As Single)

[JScript] protected override function ScaleCore(dx : float, dy : float);

Description

yyyy) ScrollControlIntoView

[C#] public void ScrollControlIntoView(Control activeControl);

1 [C++] public: void ScrollControlIntoView(Control* activeControl);

2 [VB] Public Sub ScrollControlIntoView(ByVal activeControl As Control)

3 [JScript] public function ScrollControlIntoView(activeControl : Control);

4
5 *Description*

6 Scrolls the currently active control into view if we are an AutoScroll Form that
7 has the Horiz or Vert scrollbar displayed...

8
9 *Description*

10 Scrolls the currently active control into view if we are an AutoScroll Form that
11 has the Horiz or Vert scrollbar displayed...

12 *zzzz) SetAutoScrollMargin*

13 [C#] public void SetAutoScrollMargin(int x, int y);

14 [C++] public: void SetAutoScrollMargin(int x, int y);

15 [VB] Public Sub SetAutoScrollMargin(ByVal x As Integer, ByVal y As Integer)

16 [JScript] public function SetAutoScrollMargin(x : int, y : int);

17 *Description*

18 Sets the size of the auto-scroll margins.

19 The margin sets the width and height of the border around each control. This
20 margin is used to determine when scroll bars are needed on the container and
21 where to scroll to when a control is selected. The **System.Drawing.Size.Width**
22 value. The **System.Drawing.Size.Height** value.

23 *aaaaa) SetDisplayRectLocation*

24 [C#] protected void SetDisplayRectLocation(int x, int y);

25 [C++] protected: void SetDisplayRectLocation(int x, int y);

[VB] Protected Sub SetDisplayRectLocation(ByVal x As Integer, ByVal y As Integer)

[JScript] protected function SetDisplayRectLocation(x : int, y : int);

Description

Adjusts the displayRect to be at the offset x, y. The contents of the Form is scrolled using Windows.ScrollWindowEx. X Offset (not delta!) Y Offset (not delta!)

bbbb) SetScrollState

[C#] protected void SetScrollState(int bit, bool value);

[C++] protected: void SetScrollState(int bit, bool value);

[VB] Protected Sub SetScrollState(ByVal bit As Integer, ByVal value As Boolean)

[JScript] protected function SetScrollState(bit : int, value : Boolean);

Description

Sets a given scroll state bit. The scroll state bit to set. The value to set the bit.

cccc) WndProc

[C#] protected override void WndProc(ref Message m);

[C++] protected: void WndProc(Message* m);

[VB] Overrides Protected Sub WndProc(ByRef m As Message)

[JScript] protected override function WndProc(m : Message);

1

2 *Description*

3 The button's window procedure. Inheriting classes can override this to add extra
4 functionality, but should not forget to call `base.wndProc(m)`; to ensure the
button continues to function properly. A Windows Message Object.

5 ScrollBar class (System.Windows.Forms)

6 *a) WndProc*

7

8

9 *Description*

10 Implements the basic functionality of a scroll bar control.

11 You typically do not inherit directly from **System.Windows.Forms.ScrollBar** .
12 To create your own scroll bar class, inherit from the
13 **System.Windows.Forms.VScrollBar** or
System.Windows.Forms.HScrollBar class.

14 *b) ScrollBar*

15 *Example Syntax:*

16 *c) WndProc*

17

18 [C#]	public	ScrollBar();
19 [C++]	public:	ScrollBar();
20 [VB]	Public	Sub New()
21 [JScript]	public	function ScrollBar();

22

23 *Description*

24 Initializes a new instance of the **System.Windows.Forms.ScrollBar** class.

25

By default, the **System.Windows.Forms.Control.TabStop** property is set to **false** when a new instance of a **System.Windows.Forms.ScrollBar** is created.

- d) *AccessibilityObject*
- e) *AccessibleDefaultActionDescription*
- f) *AccessibleDescription*
- g) *AccessibleName*
- h) *AccessibleRole*
- i) *AllowDrop*
- j) *Anchor*
- k) *BackColor*
- l) *WndProc*

Description

- m) *BackgroundImage*
- n) *WndProc*

```
[C#]    public    override    Image    BackgroundImage    {get;    set;}
[C++]    public:    __property    virtual    Image*    get_BackgroundImage();public:
__property    virtual    void    set_BackgroundImage(Image*);
[VB]    Overrides    Public    Property    BackgroundImage    As    Image
[JScript]    public    function    get    BackgroundImage() : Image;public    function    set
BackgroundImage(Image);
```


Description

- o) BindingContext*
- p) Bottom*
- q) Bounds*
- r) CanFocus*
- s) CanSelect*
- t) Capture*
- u) CausesValidation*
- v) ClientRectangle*
- w) ClientSize*
- x) CompanyName*
- y) Container*
- z) ContainsFocus*
- aa) ContextMenu*
- bb) Controls*
- cc) Created*
- dd) CreateParams*
- ee) WndProc*

Description

Retrieves the parameters needed to create the handle. Inheriting classes can override this to provide extra functionality. They should not, however, forget to call `base.getCreateParams()` first to get the struct filled up with the basic info.

ff) Cursor

gg) DataBindings

hh) DefaultImeMode

ii) WndProc

Description

Gets the default Input Method Editor(IME) mode supported by this control.

As implemented in the **System.Windows.Forms.ScrollBar** class, this property always returns the **System.Windows.Forms.ImeMode.Disable** value.

- jj) DefaultSize*
- kk) DesignMode*
- ll) DisplayRectangle*
- mm) Disposing*
- nn) Dock*
- oo) Enabled*
- pp) Events*
- qq) Focused*
- rr) Font*
- ss) WndProc*

Description

- tt) FontHeight*
- uu) ForeColor*
- vv) WndProc*

Description

Gets or sets the foreground color of the scroll bar control.

This property may only be changed programatically, it is not visible in the Properties window.

ww) *Handle*

xx) *HasChildren*

yy) *Height*

zz) *ImeMode*

aaa) *WndProc*

Description

Gets or sets the Input Method Editor(IME) mode supported by this control.

bbb) *InvokeRequired*

ccc) *IsAccessible*

ddd) *IsDisposed*

eee) *IsHandleCreated*

fff) *LargeChange*

ggg) *WndProc*

Description

Gets or sets a value to be added to or subtracted from to the **System.Windows.Forms.ScrollBar.Value** property when the scroll box is moved a large distance.

When the user presses the PAGE UP or PAGE DOWN key or clicks in the scroll bar track on either side of the scroll box, the **System.Windows.Forms.ScrollBar.Value** property changes according to the value set in the **System.Windows.Forms.ScrollBar.LargeChange** property.

1 *hhh) Left*

2 *iii) Location*

3 *jjj) Maximum*

4 *kkk) WndProc*

5
6
7 *Description*

8 Gets or sets the upper limit of values of the scrollable range.

9 You might consider adjusting the
10 **System.Windows.Forms.ScrollBar.Maximum** property dynamically to match
11 the size of the scroll bar's parent. This can be in proportion to pixel size or
12 number of rows or lines displayed, for example.

11 *III) Minimum*

12 *mmm) WndProc*

13
14
15 [C#] public int Minimum {get; set;}

16 [C++] public: __property int get_Minimum();public: __property void
17 set_Minimum(int);

18 [VB] Public Property Minimum As Integer

19 [JScript] public function get Minimum() : int;public function set Minimum(int);

20
21 *Description*

22 Gets or sets the lower limit of values of the scrollable range.

nnn) Name
ooo) Parent
ppp) ProductName
qqq) ProductVersion
rrr) RecreatingHandle
sss) Region
ttt) RenderRightToLeft
uuu) ResizeRedraw
vvv) Right
www) RightToLeft
xxx) ShowFocusCues
yyy) ShowKeyboardCues
zzz) Site
aaaa) Size
bbbb) SmallChange
cccc) WndProc

Description

Gets or sets the value to be added to or subtracted from to the **System.Windows.Forms.ScrollBar.Value** property when the scroll box is moved a small distance.

When the user presses one of the arrow keys or clicks one of the scroll bar buttons the **System.Windows.Forms.ScrollBar.Value** property changes according to the value set in the **System.Windows.Forms.ScrollBar.SmallChange** property.

dddd) TabIndex

eeee) TabStop

ffff) WndProc

Description

gggg) Tag

hhhh) Text

iiii) WndProc

Description

jjjj) Top

kkkk) TopLevelControl

llll) Value

mmmm) WndProc

Description

Gets or sets a numeric value that represents the current position of the scroll box on the scroll bar control.

1 *nnnn) Visible*

2 *oooo) Width*

3 *pppp) WindowTarget*

4 *qqqq) WndProc*

5 *rrrr) WndProc*

6 *ssss) WndProc*

7 *tttt) WndProc*

8 *uuuu) WndProc*

11 *Description*

12 Occurs when the scroll box has been moved by either a mouse or keyboard
13 action.

14 For more information about handling events, see .

15 *vvvv) WndProc*

18 *Description*

19 Occurs when the **System.Windows.Forms.ScrollBar.Value** property has
20 changed, either by a **System.Windows.Forms.ScrollBar.Scroll** event or
21 programmatically.

21 For more information about handling events, see .

22 *wwwv)OnEnabledChanged*

24 [C#] protected override void OnEnabledChanged(EventArgs e);

25 [C++] protected: void OnEnabledChanged(EventArgs* e);

1 [VB] Overrides Protected Sub OnEnabledChanged(ByVal e As EventArgs)

2 [JScript] protected override function OnEnabledChanged(e : EventArgs);

3 *xxxx) OnHandleCreated*

5 [C#] protected override void OnHandleCreated(EventArgs e);

6 [C++] protected: void OnHandleCreated(EventArgs* e);

7 [VB] Overrides Protected Sub OnHandleCreated(ByVal e As EventArgs)

8 [JScript] protected override function OnHandleCreated(e : EventArgs);

10 *Description*

11 Creates the handle. overridden to help set up scrollbar information.

12 *yyyy) OnScroll*

14 [C#] protected virtual void OnScroll(ScrollEventArgs se);

15 [C++] protected: virtual void OnScroll(ScrollEventArgs* se);

16 [VB] Overridable Protected Sub OnScroll(ByVal se As ScrollEventArgs)

17 [JScript] protected function OnScroll(se : ScrollEventArgs);

19 *Description*

20 Raises the **System.Windows.Forms.ScrollBar.Scroll** event.

21 Raising an event invokes the event handler through a delegate. For more
22 information, see . A **System.Windows.Forms.ScrollEventArgs** that contains
23 the event data.

zzzz) *OnValueChanged*

```
[C#]    protected    virtual    void    OnValueChanged(EventArgs    e);
[C++]    protected:    virtual    void    OnValueChanged(EventArgs*    e);
[VB]    Overridable Protected Sub OnValueChanged(ByVal e As EventArgs)
[JScript]    protected    function    OnValueChanged(e    :    EventArgs);
```

Description

Raises the **System.Windows.Forms.ScrollBar.ValueChanged** event.

Raising an event invokes the event handler through a delegate. For more information, see . An **System.EventArgs** that contains the event data.

aaaaa) *ToString*

```
[C#]            public            override            string            ToString();
[C++]            public:            String*            ToString();
[VB]    Overrides    Public    Function    ToString()    As    String
[JScript]    public    override    function    ToString()    :    String;
```

Description

bbbbbb) *UpdateScrollInfo*

```
[C#]            protected            void            UpdateScrollInfo();
[C++]            protected:            void            UpdateScrollInfo();
[VB]            Protected            Sub            UpdateScrollInfo()
[JScript]            protected            function            UpdateScrollInfo();
```

Description

Internal helper method Internal helper method

cccccc) WndProc

[C#] protected override void WndProc(ref Message m);

[C++] protected: void WndProc(Message* m);

[VB] Overrides Protected Sub WndProc(ByRef m As Message)

[JScript] protected override function WndProc(m : Message);

Description

ScrollBars enumeration (System.Windows.Forms)

a) WndProc

Description

Specifies which scroll bars will be visible on a control.

This enumeration is used by members such as
System.Windows.Forms.TextBox.ScrollBars .

b) WndProc

[C#] public const ScrollBars Both;

[C++] public: const ScrollBars Both;

[VB] Public Const Both As ScrollBars

[JScript] public var Both : ScrollBars;

Description

Both horizontal and vertical scroll bars are shown.

c) WndProc

[C#]	public	const	ScrollBars	Horizontal;
[C++]	public:	const	ScrollBars	Horizontal;
[VB]	Public	Const	Horizontal As	ScrollBars
[JScript]	public	var	Horizontal :	ScrollBars;

Description

Only horizontal scroll bars are shown.

d) WndProc

[C#]	public	const	ScrollBars	None;
[C++]	public:	const	ScrollBars	None;
[VB]	Public	Const	None As	ScrollBars
[JScript]	public	var	None :	ScrollBars;

Description

No scroll bars are shown.

e) WndProc

[C#]	public	const	ScrollBars	Vertical;
[C++]	public:	const	ScrollBars	Vertical;

[VB]	Public	Const	Vertical	As	ScrollBars
[JScript]	public	var	Vertical	:	ScrollBars;

Description

Only vertical scroll bars are shown.

ScrollButton enumeration (System.Windows.Forms)

a) ToString

Description

Specifies the type of scroll arrow to draw on a scroll bar.

This enumeration is used by members such as **System.Windows.Forms.ControlPaint.DrawScrollButton(System.Drawing.Graphics, System.Drawing.Rectangle, System.Windows.Forms.ScrollButton, System.Windows.Forms.ButtonState)** .

b) ToString

[C#]	public	const	ScrollButton	Down;
[C++]	public:	const	ScrollButton	Down;
[VB]	Public	Const	Down	As ScrollButton
[JScript]	public	var	Down	: ScrollButton;

Description

A down-scroll arrow.

c) *ToString*

[C#]	public	const	ScrollBar	Left;
[C++]	public:	const	ScrollBar	Left;
[VB]	Public	Const	Left As	ScrollBar
[JScript]	public	var	Left :	ScrollBar;

Description

A left-scroll arrow.

d) *ToString*

[C#]	public	const	ScrollBar	Max;
[C++]	public:	const	ScrollBar	Max;
[VB]	Public	Const	Max As	ScrollBar
[JScript]	public	var	Max :	ScrollBar;

Description

A maximum-scroll arrow.

e) *ToString*

[C#]	public	const	ScrollBar	Min;
[C++]	public:	const	ScrollBar	Min;
[VB]	Public	Const	Min As	ScrollBar
[JScript]	public	var	Min :	ScrollBar;

Description

A minimum-scroll arrow.

f) ToString

[C#]	public	const	ScrollBar	Right;
[C++]	public:	const	ScrollBar	Right;
[VB]	Public	Const	Right As	ScrollBar
[JScript]	public	var	Right :	ScrollBar;

Description

A right-scroll arrow.

g) ToString

[C#]	public	const	ScrollBar	Up;
[C++]	public:	const	ScrollBar	Up;
[VB]	Public	Const	Up As	ScrollBar
[JScript]	public	var	Up :	ScrollBar;

Description

An up-scroll arrow.

ScrollEventArgs class (System.Windows.Forms)

a) *ToString*

Description

Provides data for the **System.Windows.Forms.ScrollBar.Scroll** event.

The **System.Windows.Forms.ScrollBar.Scroll** event occurs when the user changes the value of the scroll bar. This event can result from a variety of actions, such as clicking a scroll bar arrow, pressing the UP ARROW or DOWN ARROW, or dragging the scroll box. The

System.Windows.Forms.ScrollEventArgs specifies the type of scroll event that occurred and the new value of the scroll bar.

b) *ScrollEventArgs*

Example Syntax:

c) *ToString*

[C#] public ScrollEventArgs(ScrollEventType type, int newValue);

[C++] public: ScrollEventArgs(ScrollEventType type, int newValue);

[VB] Public Sub New(ByVal type As ScrollEventType, ByVal newValue As Integer)

[JScript] public function ScrollEventArgs(type : ScrollEventType, newValue : int); Initializes a new instance of the **System.Windows.Forms.ScrollEventArgs** class.

Description

Initializes a new instance of the **System.Windows.Forms.ScrollEventArgs** class. One of the **System.Windows.Forms.ScrollEventType** values. The new value for the scroll bar.

d) *NewValue*

e) *ToString*

[C#] public int NewValue {get; set;}

[C++] public: __property int get_NewValue();public: __property void
set_NewValue(int);

[VB] Public Property NewValue As Integer

[JScript] public function get NewValue() : int;public function set NewValue(int);

Description

Gets or sets the new **System.Windows.Forms.ScrollBar.Value** of the scroll bar.

f) *Type*

g) *ToString*

[C#] public ScrollEventType Type {get;}

[C++] public: __property ScrollEventType get_Type();

[VB] Public ReadOnly Property Type As ScrollEventType

[JScript] public function get Type() : ScrollEventType;

Description

Gets the type of scroll event that occurred.

ScrollEventHandler delegate (System.Windows.Forms)

a) *ToString*

Description

Represents the method that handles the **Scroll** event of a **System.Windows.Forms.ScrollBar** , **System.Windows.Forms.TrackBar** , or **System.Windows.Forms.DataGrid** . The source of the event. A **System.Windows.Forms.ScrollEventArgs** that contains the event data.

When you create a **System.Windows.Forms.ScrollEventArgs** delegate, you identify the method that will handle the event. To associate the event with your event handler, add an instance of the delegate to the event. The event handler is called whenever the event occurs, unless you remove the delegate. For more information about event handler delegates, see .

ScrollEventType enumeration (System.Windows.Forms)

a) *ToString*

Description

Specifies the type of action used to raise the **System.Windows.Forms.ScrollBar.Scroll** event.

This enumeration is used by members such as **System.Windows.Forms.ScrollEventArgs.Type** .

b) *ToString*

[C#]	public	const	ScrollEventType	EndScroll;
[C++]	public:	const	ScrollEventType	EndScroll;
[VB]	Public	Const	EndScroll	As ScrollEventType
[JScript]	public	var	EndScroll	: ScrollEventType;

Description

The scroll box has stopped moving.

c) ToString

[C#]	public	const	ScrollEventType	First;
[C++]	public:	const	ScrollEventType	First;
[VB]	Public	Const	First	As ScrollEventType
[JScript]	public	var	First	: ScrollEventType;

Description

The scroll box was moved to the
System.Windows.Forms.ScrollBar.Minimum position.

d) ToString

[C#]	public	const	ScrollEventType	LargeDecrement;
[C++]	public:	const	ScrollEventType	LargeDecrement;
[VB]	Public	Const	LargeDecrement	As ScrollEventType
[JScript]	public	var	LargeDecrement	: ScrollEventType;

Description

The scroll box moved a large distance. The user clicked the scroll bar to the left(horizontal) or above(vertical) the scroll box, or pressed the PAGE UP key.

e) *ToString*

```

[C#]      public      const      ScrollEventType      LargeIncrement;
[C++]     public:     const      ScrollEventType      LargeIncrement;
[VB]      Public      Const      LargeIncrement      As      ScrollEventType
[JScript] public      var      LargeIncrement      :      ScrollEventType;

```

Description

The scroll box moved a large distance. The user clicked the scroll bar to the right(horizontal) or below(vertical) the scroll box, or pressed the PAGE DOWN key.

f) *ToString*

```

[C#]      public      const      ScrollEventType      Last;
[C++]     public:     const      ScrollEventType      Last;
[VB]      Public      Const      Last      As      ScrollEventType
[JScript] public      var      Last      :      ScrollEventType;

```

Description

The scroll box was moved to the **System.Windows.Forms.ScrollBar.Maximum** position.

g) *ToString*

```

[C#]      public      const      ScrollEventType      SmallDecrement;
[C++]     public:     const      ScrollEventType      SmallDecrement;
[VB]      Public      Const      SmallDecrement      As      ScrollEventType

```

1 [JScript] public var SmallDecrement : ScrollEventType;

2
3 *Description*

4 The scroll box was moved a small distance. The user clicked the left(horizontal)
5 or top(vertical) scroll arrow or pressed the UP ARROW key.

6 *h) ToString*

7 [C#] public const ScrollEventType SmallIncrement;

8 [C++] public: const ScrollEventType SmallIncrement;

9 [VB] Public Const SmallIncrement As ScrollEventType

10 [JScript] public var SmallIncrement : ScrollEventType;

11
12 *Description*

13 The scroll box was moved a small distance. The user clicked the right(horizontal)
14 or bottom(vertical) scroll arrow or pressed the DOWN ARROW key.

15 *i) ToString*

16
17 [C#] public const ScrollEventType ThumbPosition;

18 [C++] public: const ScrollEventType ThumbPosition;

19 [VB] Public Const ThumbPosition As ScrollEventType

20 [JScript] public var ThumbPosition : ScrollEventType;

21
22 *Description*

23 The scroll box was moved.
24
25

j) ToString

[C#]	public	const	ScrollEventType	ThumbTrack;
[C++]	public:	const	ScrollEventType	ThumbTrack;
[VB]	Public	Const	ThumbTrack	As ScrollEventType
[JScript]	public	var	ThumbTrack	: ScrollEventType;

Description

The scroll box is currently being moved.

SecurityIDType enumeration (System.Windows.Forms)

a) ToString

Description

b) ToString

[C#]	public	const	SecurityIDType	Alias;
[C++]	public:	const	SecurityIDType	Alias;
[VB]	Public	Const	Alias	As SecurityIDType
[JScript]	public	var	Alias	: SecurityIDType;

Description

c) *ToString*

[C#]	public	const	SecurityIDType	Computer;
[C++]	public:	const	SecurityIDType	Computer;
[VB]	Public	Const	Computer	As SecurityIDType
[JScript]	public	var	Computer	: SecurityIDType;

Description

d) *ToString*

[C#]	public	const	SecurityIDType	DeletedAccount;
[C++]	public:	const	SecurityIDType	DeletedAccount;
[VB]	Public	Const	DeletedAccount	As SecurityIDType
[JScript]	public	var	DeletedAccount	: SecurityIDType;

Description

e) *ToString*

[C#]	public	const	SecurityIDType	Domain;
[C++]	public:	const	SecurityIDType	Domain;
[VB]	Public	Const	Domain	As SecurityIDType
[JScript]	public	var	Domain	: SecurityIDType;

Description

f) ToString

[C#]	public	const	SecurityIDType	Group;
[C++]	public:	const	SecurityIDType	Group;
[VB]	Public	Const	Group As	SecurityIDType
[JScript]	public	var	Group :	SecurityIDType;

Description

g) ToString

[C#]	public	const	SecurityIDType	Invalid;
[C++]	public:	const	SecurityIDType	Invalid;
[VB]	Public	Const	Invalid As	SecurityIDType
[JScript]	public	var	Invalid :	SecurityIDType;

Description

h) ToString

[C#]	public	const	SecurityIDType	Unknown;
[C++]	public:	const	SecurityIDType	Unknown;

[VB]	Public	Const	Unknown	As	SecurityIDType
[JScript]	public	var	Unknown	:	SecurityIDType;

Description

i) ToString

[C#]	public	const	SecurityIDType	User;	
[C++]	public:	const	SecurityIDType	User;	
[VB]	Public	Const	User	As	SecurityIDType
[JScript]	public	var	User	:	SecurityIDType;

Description

j) ToString

[C#]	public	const	SecurityIDType	WellKnownGroup;	
[C++]	public:	const	SecurityIDType	WellKnownGroup;	
[VB]	Public	Const	WellKnownGroup	As	SecurityIDType
[JScript]	public	var	WellKnownGroup	:	SecurityIDType;

Description

1 SelectedGridItemChangedEventArgs class (System.Windows.Forms)

2 a) *ToString*

3
4
5 *Description*

6 Provides data for the
7 **System.Windows.Forms.PropertyGrid.SelectedGridItemChanged** event
8 of the **System.Windows.Forms.PropertyGrid** control.

9 The **System.Windows.Forms.PropertyGrid.SelectedGridItemChanged**
10 event occurs when the user changes the **System.Windows.Forms.GridItem**
11 that is selected in a **System.Windows.Forms.PropertyGrid** .

12 b) *SelectedGridItemChangedEventArgs*

13 *Example Syntax:*

14 c) *ToString*

15 [C#] public SelectedGridItemChangedEventArgs(GridItem oldSel, GridItem
16 newSel);

17 [C++] public: SelectedGridItemChangedEventArgs(GridItem* oldSel, GridItem*
18 newSel);

19 [VB] Public Sub New(ByVal oldSel As GridItem, ByVal newSel As GridItem)

20 [JScript] public function SelectedGridItemChangedEventArgs(oldSel : GridItem,
21 newSel : GridItem);

22 *Description*

23 Initializes a new instance of the
24 **System.Windows.Forms.SelectedGridItemChangedEventArgs** class. The
25 previously selected grid item. The newly selected grid item.

d) *NewSelection*

e) *ToString*

```
[C#]      public      GridItem      NewSelection      {get;}
[C++]      public:      __property      GridItem*      get_NewSelection();
[VB]      Public      ReadOnly      Property      NewSelection      As      GridItem
[JScript]      public      function      get      NewSelection()      :      GridItem;
```

Description

Gets the newly selected **System.Windows.Forms.GridItem** object.

f) *OldSelection*

g) *ToString*

```
[C#]      public      GridItem      OldSelection      {get;}
[C++]      public:      __property      GridItem*      get_OldSelection();
[VB]      Public      ReadOnly      Property      OldSelection      As      GridItem
[JScript]      public      function      get      OldSelection()      :      GridItem;
```

Description

Gets the previously selected **System.Windows.Forms.GridItem** object.

delegate

(System.Windows.Forms)

a) *ToString*

Description

Represents the method that will handle the **System.Windows.Forms.PropertyGrid.SelectedGridItemChanged** event of a **System.Windows.Forms.PropertyGrid** . The source of the event. A **System.Windows.Forms.SelectedGridItemChangedEventArgs** that contains the event data.

When you create a **System.Windows.Forms.SelectedGridItemChangedEventHandler** delegate, you identify the method that will handle the event. To associate the event with your event handler, add an instance of the delegate to the event. The event handler is called whenever the event occurs, unless you remove the delegate.

ListBox.SelectedIndexCollection class (System.Windows.Forms)

a) *ToString*

Description

Represents the collection containing the indices to the selected items in a **System.Windows.Forms.ListBox** .

The **System.Windows.Forms.ListBox.SelectedIndexCollection** class stores the indices to the selected items in the **System.Windows.Forms.ListBox** .

The indices stored in the **System.Windows.Forms.ListBox.SelectedIndexCollection** are index positions within the **System.Windows.Forms.ListBox.ObjectCollection** class. The **System.Windows.Forms.ListBox.ObjectCollection** class stores all items displayed in the **System.Windows.Forms.ListBox** .

b) *ListBox.SelectedIndexCollection*

Example Syntax:

c) *ToString*

```
[C#]      public      ListBox.SelectedIndexCollection(ListBox      owner);
[C++]      public:      SelectedIndexCollection(ListBox*      owner);
[VB]      Public      Sub      New(ByVal      owner      As      ListBox)
[JScript] public function ListBox.SelectedIndexCollection(owner : ListBox);
```

Description

Initializes a new instance of the **System.Windows.Forms.ListBox.SelectedIndexCollection** class.

You cannot create an instance of this class without associating it with a **System.Windows.Forms.ListBox** control. A **System.Windows.Forms.ListBox** representing the owner of the collection.

d) *Count*

e) *ToString*

```
[C#]      public      int      Count      {get;}
[C++]      public:      __property      int      get_Count();
[VB]      Public      ReadOnly      Property      Count      As      Integer
[JScript] public      function      get      Count()      :      int;
```

Description

Gets the number of items in the collection.

This property enables you to determine the number of selected items in the **System.Windows.Forms.ListBox**. You can then use this value when looping

through the values of the collection and you need to provide a number of iterations to perform the loop. Unless the **System.Windows.Forms.ListBox.SelectionMode** property of the **System.Windows.Forms.ListBox** is set to **SelectionMode.MultiSimple** or **SelectionMode.MultiExtended**, this property always returns a value of zero (0) or one (1) depending on whether you have a selected item.

f) *IsReadOnly*

g) *ToString*

[C#]	public	bool	IsReadOnly	{get;}
[C++]	public:	__property	bool	get_IsReadOnly();
[VB]	Public	ReadOnly	Property	IsReadOnly As Boolean
[JScript]	public	function	get_IsReadOnly()	: Boolean;

Description

Gets a value indicating whether the collection is read-only.

This property is always **true** for this collection. The items in this collection are modified only by the **System.Windows.Forms.ListBox** control.

h) *Item*

i) *ToString*

[C#]	public	int	this[int index]	{get;}
[C++]	public:	__property	int	get_Item(int index);
[VB]	Public	Default	ReadOnly	Property Item(ByVal index As Integer) As Integer
[JScript]	returnValue	=	SelectedIndexCollectionObject.Item(index);	

Description

Gets the index value at the specified index within this collection.

This indexer enables you to get a specific selected index from the **System.Windows.Forms.ListBox.SelectedIndexCollection** . The index stored in the collection is an index into the **System.Windows.Forms.ListBox.ObjectCollection** of the **System.Windows.Forms.ListBox** that represents a selected item in the **System.Windows.Forms.ListBox** . The index of the item in the collection to get.

j) Contains

```
[C#]      public      bool      Contains(int      selectedIndex);
[C++]     public:     bool      Contains(int      selectedIndex);
[VB]      Public Function Contains(ByVal selectedIndex As Integer) As Boolean
[JScript] public  function  Contains(selectedIndex : int) : Boolean;
```

Description

Determines whether the specified index is located within the collection.

Return Value: **true** if the specified index from the

System.Windows.Forms.ListBox.ObjectCollection for the **System.Windows.Forms.ListBox** is an item in this collection; otherwise, **false** .

The

System.Windows.Forms.ListBox.SelectedIndexCollection.Contains(System.Int32) method enables you to determine whether an index position from the **System.Windows.Forms.ListBox.ObjectCollection** class is a member of the selected indices stored in the

System.Windows.Forms.ListBox.SelectedIndexCollection . You can use this to determine whether a specific item in a multiple-selection **System.Windows.Forms.ListBox** is selected. The index to locate in the collection.

k) CopyTo

```
[C#]      public      void      CopyTo(Array      dest,      int      index);
[C++]     public:     __sealed void      CopyTo(Array*      dest,      int      index);
```

1 [VB] NotOverridable Public Sub CopyTo(ByVal dest As Array, ByVal index As
2 Integer)

3 [JScript] public function CopyTo(dest : Array, index : int);

4
5 *Description*

6 Copies the entire collection into an existing array at a specified location within
the array.

7 You can use this method to combine the selected indexes from multiple
8 collections into a single array. The destination array. The index in the destination
array at which storing begins.

9
10 *l) GetEnumerator*

11 [C#] public IEnumerator GetEnumerator();

12 [C++] public: __sealed IEnumerator* GetEnumerator();

13 [VB] NotOverridable Public Function GetEnumerator() As IEnumerator

14 [JScript] public function GetEnumerator() : IEnumerator;

15
16 *Description*

17 Returns an enumerator to use to iterate through the selected indices collection.

18 *Return Value:* An **System.Collections.IEnumerator** object that represents the
selected indices collection.

19
20 *m) IndexOf*

21
22 [C#] public int IndexOf(int selectedIndex);

23 [C++] public: int IndexOf(int selectedIndex);

24 [VB] Public Function IndexOf(ByVal selectedIndex As Integer) As Integer

25 [JScript] public function IndexOf(selectedIndex : int) : int;

Description

Returns the index within the

System.Windows.Forms.ListBox.SelectedIndexCollection of the specified index from the **System.Windows.Forms.ListBox.ObjectCollection** of the **System.Windows.Forms.ListBox** .

Return Value: The zero-based index in the collection where the specified index of the **System.Windows.Forms.ListBox.ObjectCollection** was located within the **System.Windows.Forms.ListBox.SelectedIndexCollection** ; otherwise, negative one (-1).

Once you know that an item is located within the collection (using the

System.Windows.Forms.ListBox.SelectedIndexCollection.Contains(System.Int32) method), you can use the

System.Windows.Forms.ListBox.SelectedIndexCollection.IndexOf(System.Int32) method to determine where a specific index position within the **System.Windows.Forms.ListBox.ObjectCollection** for the **System.Windows.Forms.ListBox** is stored within the **System.Windows.Forms.ListBox.SelectedIndexCollection** . The zero-based index from the **System.Windows.Forms.ListBox.ObjectCollection** to locate in this collection.

n) *IList.Add*

[C#] int IList.Add(object value);

[C++] int IList::Add(Object* value);

[VB] Function Add(ByVal value As Object) As Integer Implements IList.Add

[JScript] function IList.Add(value : Object) : int;

o) *IList.Clear*

[C#] void IList.Clear();

[C++] void IList::Clear();

[VB] Sub Clear() Implements IList.Clear

[JScript] function IList.Clear();

p) *IList.Contains*

[C#] bool IList.Contains(object selectedIndex);

[C++] bool IList::Contains(Object* selectedIndex);

[VB] Function Contains(ByVal selectedIndex As Object) As Boolean Implements
IList.Contains

[JScript] function IList.Contains(selectedIndex : Object) : Boolean;

q) *IList.IndexOf*

[C#] int IList.IndexOf(object selectedIndex);

[C++] int IList::IndexOf(Object* selectedIndex);

[VB] Function IndexOf(ByVal selectedIndex As Object) As Integer Implements
IList.IndexOf

[JScript] function IList.IndexOf(selectedIndex : Object) : int;

r) *IList.Insert*

[C#] void IList.Insert(int index, object value);

[C++] void IList::Insert(int index, Object* value);

[VB] Sub Insert(ByVal index As Integer, ByVal value As Object) Implements
IList.Insert

[JScript] function IList.Insert(index : int, value : Object);

s) *IList.Remove*

[C#] void IList.Remove(object value);

[C++] void IList::Remove(Object* value);

1 [VB] Sub Remove(ByVal value As Object) Implements IList.Remove

2 [JScript] function IList.Remove(value : Object);

3 t) *IList.RemoveAt*

5 [C#] void IList.RemoveAt(int index);

6 [C++] void IList::RemoveAt(int index);

7 [VB] Sub RemoveAt(ByVal index As Integer) Implements IList.RemoveAt

8 [JScript] function IList.RemoveAt(index : int);

9 ListView.SelectedIndexCollection class (System.Windows.Forms)

10 a) *ToString*

13 *Description*

15 b) *ListView.SelectedIndexCollection*

16 *Example Syntax:*

17 c) *ToString*

19 [C#] public ListView.SelectedIndexCollection(ListView owner);

20 [C++] public: SelectedIndexCollection(ListView* owner);

21 [VB] Public Sub New(ByVal owner As ListView)

22 [JScript] public function ListView.SelectedIndexCollection(owner : ListView);

24 *Description*

d) *Count*

e) *ToString*

[C#] public int Count {get;}

[C++] public: __property int get_Count();

[VB] Public ReadOnly Property Count As Integer

[JScript] public function get Count() : int;

Description

Gets the number of currently selected items.

f) *IsReadOnly*

g) *ToString*

[C#] public bool IsReadOnly {get;}

[C++] public: __property bool get_IsReadOnly();

[VB] Public ReadOnly Property IsReadOnly As Boolean

[JScript] public function get IsReadOnly() : Boolean;

Description

Gets whether the collection is read-only.

h) *Item*

i) *ToString*

[C#] public int this[int index] {get;}

[C++] public: __property int get_Item(int index);

1 [VB] Public Default ReadOnly Property Item(ByVal index As Integer) As Integer

2 [JScript] returnValue = SelectedIndexCollectionObject.Item(index);

3
4 *Description*

5 Selected item in the list.

6 *j) Contains*

7
8 [C#] public bool Contains(int selectedIndex);

9 [C++] public: bool Contains(int selectedIndex);

10 [VB] Public Function Contains(ByVal selectedIndex As Integer) As Boolean

11 [JScript] public function Contains(selectedIndex : int) : Boolean;

12
13 *Description*

14
15 *k) CopyTo*

16
17 [C#] public void CopyTo(Array dest, int index);

18 [C++] public: __sealed void CopyTo(Array* dest, int index);

19 [VB] NotOverridable Public Sub CopyTo(ByVal dest As Array, ByVal index As
20 Integer)

21 [JScript] public function CopyTo(dest : Array, index : int);

22
23 *Description*

l) GetEnumerator

```
[C#]          public          IEnumerator          GetEnumerator();
[C++]          public:          __sealed          IEnumerator*          GetEnumerator();
[VB] NotOverridable Public Function GetEnumerator() As IEnumerator
[JavaScript]    public          function          GetEnumerator()          :          IEnumerator;
```

Description

m) IndexOf

```
[C#]          public          int          IndexOf(int          selectedIndex);
[C++]          public:          int          IndexOf(int          selectedIndex);
[VB] Public Function IndexOf(ByVal selectedIndex As Integer) As Integer
[JavaScript]    public          function          IndexOf(selectedIndex          :          int)          :          int;
```

Description

n) IList.Add

```
[C#]          int          IList.Add(object          value);
[C++]          int          IList::Add(Object*          value);
[VB] Function Add(ByVal value As Object) As Integer Implements IList.Add
[JavaScript]    function IList.Add(value : Object) : int;
```

o) IList.Clear

[C#] void IList.Clear();
 [C++] void IList::Clear();
 [VB] Sub Clear() Implements IList.Clear
 [JScript] function IList.Clear();

p) IList.Contains

[C#] bool IList.Contains(object selectedIndex);
 [C++] bool IList::Contains(Object* selectedIndex);
 [VB] Function Contains(ByVal selectedIndex As Object) As Boolean Implements
 IList.Contains
 [JScript] function IList.Contains(selectedIndex : Object) : Boolean;

q) IList.IndexOf

[C#] int IList.IndexOf(object selectedIndex);
 [C++] int IList::IndexOf(Object* selectedIndex);
 [VB] Function IndexOf(ByVal selectedIndex As Object) As Integer Implements
 IList.IndexOf
 [JScript] function IList.IndexOf(selectedIndex : Object) : int;

r) IList.Insert

[C#] void IList.Insert(int index, object value);
 [C++] void IList::Insert(int index, Object* value);
 [VB] Sub Insert(ByVal index As Integer, ByVal value As Object) Implements

1 IList.Insert

2 [JScript] function IList.Insert(index : int, value : Object);

3 s) *IList.Remove*

5 [C#] void IList.Remove(object value);

6 [C++] void IList::Remove(Object* value);

7 [VB] Sub Remove(ByVal value As Object) Implements IList.Remove

8 [JScript] function IList.Remove(value : Object);

9 t) *IList.RemoveAt*

11 [C#] void IList.RemoveAt(int index);

12 [C++] void IList::RemoveAt(int index);

13 [VB] Sub RemoveAt(ByVal index As Integer) Implements IList.RemoveAt

14 [JScript] function IList.RemoveAt(index : int);

15 ListView.SelectedListViewItemCollection class (System.Windows.Forms)

16 a) *ToString*

19 *Description*

21 b) *ListView.SelectedListViewItemCollection*

22 *Example Syntax:*

c) *ToString*

```
[C#]    public    ListView.SelectedListViewItemCollection(ListView    owner);
[C++]    public:    SelectedListViewItemCollection(ListView*    owner);
[VB]    Public    Sub    New(ByVal    owner    As    ListView)
[JScript] public function ListView.SelectedListViewItemCollection(owner :
ListView);
```

Description

d) *Count*

e) *ToString*

```
[C#]                public                int                Count                {get;}
[C++]                public:                __property                int                get_Count();
[VB]    Public    ReadOnly    Property    Count    As    Integer
[JScript]    public    function    get    Count()    :    int;
```

Description

Gets the number of currently selected items.

f) *IsReadOnly*

g) *ToString*

```
[C#]                public                bool                IsReadOnly                {get;}
[C++]                public:                __property                bool                get_IsReadOnly();
```

[VB] Public ReadOnly Property IsReadOnly As Boolean
 [JScript] public function get IsReadOnly() : Boolean;

Description

h) Item

i) ToString

[C#] public ListViewItem this[int index] {get;}
 [C++] public: __property ListViewItem* get_Item(int index);
 [VB] Public Default ReadOnly Property Item(ByVal index As Integer) As
 ListViewItem
 [JScript] returnValue = SelectedListViewItemCollectionObject.Item(index);

Description

Selected item in the list.

j) Clear

[C#] public void Clear();
 [C++] public: __sealed void Clear();
 [VB] NotOverridable Public Sub Clear()
 [JScript] public function Clear();

Description

Unselects all items.

k) Contains

[C#] public bool Contains(ListViewItem item);
[C++] public: bool Contains(ListViewItem* item);
[VB] Public Function Contains(ByVal item As ListViewItem) As Boolean
[JScript] public function Contains(item : ListViewItem) : Boolean;

Description

l) CopyTo

[C#] public void CopyTo(Array dest, int index);
[C++] public: __sealed void CopyTo(Array* dest, int index);
[VB] NotOverridable Public Sub CopyTo(ByVal dest As Array, ByVal index As Integer)
[JScript] public function CopyTo(dest : Array, index : int);

Description

m) GetEnumerator

[C#] public IEnumerator GetEnumerator();
[C++] public: __sealed IEnumerator* GetEnumerator();
[VB] NotOverridable Public Function GetEnumerator() As IEnumerator
[JScript] public function GetEnumerator() : IEnumerator;

1
2 *Description*

3
4 *n) IndexOf*

5
6 [C#] public int IndexOf(ListViewItem item);

7 [C++] public: int IndexOf(ListViewItem* item);

8 [VB] Public Function IndexOf(ByVal item As ListViewItem) As Integer

9 [JScript] public function IndexOf(item : ListViewItem) : int;

10
11 *Description*

12
13 *o) IList.Add*

14
15 [C#] int IList.Add(object value);

16 [C++] int IList::Add(Object* value);

17 [VB] Function Add(ByVal value As Object) As Integer Implements IList.Add

18 [JScript] function IList.Add(value : Object) : int;

19 *p) IList.Contains*

20
21 [C#] bool IList.Contains(object item);

22 [C++] bool IList::Contains(Object* item);

23 [VB] Function Contains(ByVal item As Object) As Boolean Implements

24

25

1 IList.Contains

2 [JScript] function IList.Contains(item : Object) : Boolean;

3 *q) IList.IndexOf*

4
5 [C#] int IList.IndexOf(object item);

6 [C++] int IList::IndexOf(Object* item);

7 [VB] Function IndexOf(ByVal item As Object) As Integer Implements
8 IList.IndexOf

9 [JScript] function IList.IndexOf(item : Object) : int;

10 *r) IList.Insert*

11
12 [C#] void IList.Insert(int index, object value);

13 [C++] void IList::Insert(int index, Object* value);

14 [VB] Sub Insert(ByVal index As Integer, ByVal value As Object) Implements
15 IList.Insert

16 [JScript] function IList.Insert(index : int, value : Object);

17 *s) IList.Remove*

18
19 [C#] void IList.Remove(object value);

20 [C++] void IList::Remove(Object* value);

21 [VB] Sub Remove(ByVal value As Object) Implements IList.Remove

22 [JScript] function IList.Remove(value : Object);

t) *IList.RemoveAt*

```
[C#]          void          IList.RemoveAt(int          index);
[C++]          void          IList::RemoveAt(int          index);
[VB] Sub RemoveAt(ByVal index As Integer) Implements IList.RemoveAt
[JScript] function IList.RemoveAt(index : int);

    ListBox.SelectedObjectCollection class (System.Windows.Forms)
```

a) *ToString*

Description

Represents the collection of selected items in the **System.Windows.Forms.ListBox** .

The **System.Windows.Forms.ListBox.SelectedObjectCollection** class stores the selected items in the **System.Windows.Forms.ListBox** . The items stored in the **System.Windows.Forms.ListBox.SelectedObjectCollection** are items contained within the **System.Windows.Forms.ListBox.ObjectCollection** class. The **System.Windows.Forms.ListBox.ObjectCollection** class stores all items displayed in the **System.Windows.Forms.ListBox** .

b) *ListBox.SelectedObjectCollection*

Example Syntax:

c) *ToString*

```
[C#]          public          ListBox.SelectedObjectCollection(ListBox          owner);
[C++]          public:          SelectedObjectCollection(ListBox*          owner);
[VB]          Public          Sub          New(ByVal          owner          As          ListBox)
[JScript] public function ListBox.SelectedObjectCollection(owner : ListBox);
```

Description

Initializes a new instance of the **System.Windows.Forms.ListBox.SelectedObjectCollection** class.

An instance of this class cannot be created without associating it with a **System.Windows.Forms.ListBox** control. A **System.Windows.Forms.ListBox** representing the owner of the collection.

d) *Count*

e) *ToString*

[C#]	public	int	Count	{get;}
[C++]	public:	__property	int	get_Count();
[VB]	Public	ReadOnly	Property	Count As Integer
[JScript]	public	function	get	Count() : int;

Description

Gets the number of items in the collection.

This property enables you to determine the number of selected items in the **System.Windows.Forms.ListBox**. You can then use this value when looping through the values of the collection and you need to provide a number of iterations to perform the loop. Unless the **System.Windows.Forms.ListBox.SelectionMode** property of the **System.Windows.Forms.ListBox** is set to **SelectionMode.MultiSimple** or **SelectionMode.MultiExtended**, this property always returns a value of zero (0) or one (1) depending on whether you have a selected item.

f) *IsReadOnly*

g) *ToString*

[C#]	public	bool	IsReadOnly	{get;}
------	--------	------	------------	--------

```

1 [C++]      public:      __property      bool      get_IsReadOnly();
2 [VB]      Public      ReadOnly      Property      IsReadOnly      As      Boolean
3 [JScript]      public      function      get      IsReadOnly()      :      Boolean;

```

Description

Gets a value indicating whether the collection is read-only.

This property is always **true** for this collection.

h) Item

i) ToString

```

11 [C#]      public      object      this[int      index]      {get;      set;}
12 [C++]      public:      __property      Object*      get_Item(int      index);public:      __property      void
13      set_Item(int      index,      Object*);
14 [VB]      Public      Default      Property      Item(ByVal      index      As      Integer)      As      Object
15 [JScript]      returnValue      =
16      SelectedObjectCollectionObject.Item(index);SelectedObjectCollectionObject.Item
17      (index)      =      returnValue;

```

Description

Gets the item at the specified index within the collection.

This indexer enables you to get a specific item from the **System.Windows.Forms.ListBox.SelectedObjectCollection** . The item stored in this collection is an item within the **System.Windows.Forms.ListBox.ObjectCollection** of the **System.Windows.Forms.ListBox** that represents a selected item in the **System.Windows.Forms.ListBox** . The index of the item in the collection to retrieve.

j) *Contains*

```
[C#]      public      bool      Contains(object      selectedObject);  
[C++]    public:    __sealed    bool    Contains(Object*    selectedObject);  
[VB] NotOverridable Public Function Contains(ByVal selectedObject As Object)  
As                                             Boolean  
[JScript] public function Contains(selectedObject : Object) : Boolean;
```

Description

Determines whether the specified item is located within the collection.

Return Value: **true** if the specified item is located in the collection; otherwise, **false**.

The

System.Windows.Forms.ListBox.SelectedObjectCollection.Contains(System.Object) method enables you to determine whether an item from the **System.Windows.Forms.ListBox.ObjectCollection** class is a member of the selected items stored in the **System.Windows.Forms.ListBox.SelectedObjectCollection**. You can use this to determine if a specific item in a multiple-selection **System.Windows.Forms.ListBox** is selected. An object representing the item to locate in the collection.

k) *CopyTo*

```
[C#]      public      void      CopyTo(Array      dest,      int      index);  
[C++]    public:    __sealed    void    CopyTo(Array*    dest,      int      index);  
[VB] NotOverridable Public Sub CopyTo(ByVal dest As Array, ByVal index As  
Integer)  
[JScript] public function CopyTo(dest : Array, index : int);
```

Description

Copies the entire collection into an existing array at a specified location within the array.

You can use this method to combine the selected indexes from multiple collections into a single array. An **System.Array** representing the array to copy the contents of the collection to. The location within the destination array to copy the items from the collection to.

l) GetEnumerator

```
[C#]          public          IEnumerator          GetEnumerator();
[C++]        public:    __sealed    IEnumerator*    GetEnumerator();
[VB] NotOverridable Public Function GetEnumerator() As IEnumerator
[JScript]    public    function    GetEnumerator()    :    IEnumerator;
```

Description

Returns an enumerator that can be used to iterate through the selected item collection.

Return Value: An **System.Collections.IEnumerator** object that represents the item collection.

m) IndexOf

```
[C#]          public          int          IndexOf(object          selectedObject);
[C++]        public:    __sealed    int    IndexOf(Object*    selectedObject);
[VB] NotOverridable Public Function IndexOf(ByVal selectedObject As Object)
As
Integer
[JScript]    public    function    IndexOf(selectedObject    :    Object)    :    int;
```

Description

Returns the index within the collection of the specified item.

Return Value: The zero-based index of the item in the collection; otherwise, -1.

Once you know that an item is located within the collection (using the **System.Windows.Forms.ListBox.SelectedObjectCollection.Contains(System.Object)** method), you can use the **System.Windows.Forms.ListBox.SelectedObjectCollection.IndexOf(System.Object)** method to determine where a specific item within the **System.Windows.Forms.ListBox.ObjectCollection** for the **System.Windows.Forms.ListBox** is stored within the **System.Windows.Forms.ListBox.SelectedObjectCollection**. An object representing the item to locate in the collection.

n) *IList.Add*

[C#] int IList.Add(object value);

[C++] int IList::Add(Object* value);

[VB] Function Add(ByVal value As Object) As Integer Implements IList.Add

[JScript] function IList.Add(value : Object) : int;

o) *IList.Clear*

[C#] void IList.Clear();

[C++] void IList::Clear();

[VB] Sub Clear() Implements IList.Clear

[JScript] function IList.Clear();

p) *IList.Insert*

[C#] void IList.Insert(int index, object value);

1 [C++] void IList::Insert(int index, Object* value);

2 [VB] Sub Insert(ByVal index As Integer, ByVal value As Object) Implements

3 IList.Insert

4 [JScript] function IList.Insert(index : int, value : Object);

5 *q) IList.Remove*

7 [C#] void IList.Remove(object value);

8 [C++] void IList::Remove(Object* value);

9 [VB] Sub Remove(ByVal value As Object) Implements IList.Remove

10 [JScript] function IList.Remove(value : Object);

11 *r) IList.RemoveAt*

13 [C#] void IList.RemoveAt(int index);

14 [C++] void IList::RemoveAt(int index);

15 [VB] Sub RemoveAt(ByVal index As Integer) Implements IList.RemoveAt

16 [JScript] function IList.RemoveAt(index : int);

17 SelectionMode enumeration (System.Windows.Forms)

18 *a) ToString*

21 *Description*

22 Specifies the selection behavior of a list box.

23 This enumeration is used by classes such as **System.Windows.Forms.ListBox**
24 and **System.Windows.Forms.CheckedListBox** .

b) ToString

[C#]	public	const	SelectionMode	MultiExtended;
[C++]	public:	const	SelectionMode	MultiExtended;
[VB]	Public	Const	MultiExtended	As SelectionMode
[JScript]	public	var	MultiExtended	: SelectionMode;

Description

Multiple items can be selected, and the user can use the SHIFT, CTRL, and arrow keys to make selections. Indicates that more than one item at a time can be selected, and keyboard combinations, such as SHIFT and CTRL can be used to help in selection.

c) ToString

[C#]	public	const	SelectionMode	MultiSimple;
[C++]	public:	const	SelectionMode	MultiSimple;
[VB]	Public	Const	MultiSimple	As SelectionMode
[JScript]	public	var	MultiSimple	: SelectionMode;

Description

Multiple items can be selected.

d) ToString

[C#]	public	const	SelectionMode	None;
[C++]	public:	const	SelectionMode	None;
[VB]	Public	Const	None	As SelectionMode

1 [JScript] public var None : SelectionMode;

3 *Description*

4 No items can be selected.

5 e) *ToString*

7 [C#] public const SelectionMode One;

8 [C++] public: const SelectionMode One;

9 [VB] Public Const One As SelectionMode

10 [JScript] public var One : SelectionMode;

12 *Description*

13 Only one item can be selected.

14 SelectionRange class (System.Windows.Forms)

15 a) *ToString*

18 *Description*

19 Represents a date selection range in a month calendar control.

20 The **System.Windows.Forms.SelectionRange** is the date or dates selected
21 and highlighted on the **System.Windows.Forms.MonthCalendar** control. If
only one date is selected, the

System.Windows.Forms.SelectionRange.Start and

System.Windows.Forms.SelectionRange.End property values will be equal.

23 The **System.Windows.Forms.SelectionRange** may be changed by the user
clicking a date while dragging the mouse pointer across the desired dates, or you
24 can set the range in code. For example, you may want to have the user enter a
date range into two **System.Windows.Forms.TextBox** controls or two

System.Windows.Forms.DateTimePicker controls and set the **System.Windows.Forms.SelectionRange** based on those dates.

b) SelectionRange

Example Syntax:

c) ToString

```
[C#] public SelectionRange();
[C++] public: SelectionRange();
[VB] Public Sub New()
[JavaScript] public function SelectionRange(); Initializes a new instance of the
System.Windows.Forms.SelectionRange class.
```

Description

Initializes a new instance of the **System.Windows.Forms.SelectionRange** class.

The **System.Windows.Forms.SelectionRange.Start** and **System.Windows.Forms.SelectionRange.End** values are set to null when this constructor is used.

d) SelectionRange

Example Syntax:

e) ToString

```
[C#] public SelectionRange(SelectionRange range);
[C++] public: SelectionRange(SelectionRange* range);
[VB] Public Sub New(ByVal range As SelectionRange)
[JavaScript] public function SelectionRange(range : SelectionRange);
```

Description

Initializes a new instance of the **System.Windows.Forms.SelectionRange** class with the specified selection range.

The **System.Windows.Forms.SelectionRange.Start** and **System.Windows.Forms.SelectionRange.End** property values are assigned the **System.Windows.Forms.SelectionRange.Start** and **System.Windows.Forms.SelectionRange.End** property values of the assigned **System.Windows.Forms.SelectionRange** object. The existing **System.Windows.Forms.SelectionRange** object.

f) *SelectionRange*

Example Syntax:

g) *ToString*

```
[C#] public SelectionRange(DateTime lower, DateTime upper);
```

```
[C++] public: SelectionRange(DateTime lower, DateTime upper);
```

```
[VB] Public Sub New(ByVal lower As DateTime, ByVal upper As DateTime)
```

```
[JScript] public function SelectionRange(lower : DateTime, upper : DateTime);
```

Description

Initializes a new instance of the **System.Windows.Forms.SelectionRange** class with the specified beginning and ending dates.

If the *lower* **System.DateTime** value is greater than the *upper* **System.DateTime** value, the *lower* value will be assigned to the **System.Windows.Forms.SelectionRange.End** property instead of the **System.Windows.Forms.SelectionRange.Start** property. The starting date in the **System.Windows.Forms.SelectionRange** . The ending date in the **System.Windows.Forms.SelectionRange** .

h) *End*

i) *ToString*

[C#] public DateTime End {get; set;}

[C++] public: __property DateTime get_End();public: __property void
set_End(DateTime);

[VB] Public Property End As DateTime

[JScript] public function get End() : DateTime;public function set End(DateTime);

Description

Gets or sets the ending date and time of the selection range.

j) *Start*

k) *ToString*

[C#] public DateTime Start {get; set;}

[C++] public: __property DateTime get_Start();public: __property void
set_Start(DateTime);

[VB] Public Property Start As DateTime

[JScript] public function get Start() : DateTime;public function set
Start(DateTime);

Description

Gets or sets the starting date and time of the selection range.

1) *ToString*

[C#]	public	override	string	ToString();
[C++]	public:		String*	ToString();
[VB]	Overrides	Public	Function	ToString() As String
[JScript]	public	override	function	ToString() : String;

Description

Returns a string representation for this control.

Return Value: String Returns a string representation for this control.

SelectionRangeConverter class (System.Windows.Forms)

a) *ToString*

Description

Provides a type converter to convert

System.Windows.Forms.SelectionRange objects to and from various other representations.

For more information about type converters, see the

System.ComponentModel.TypeConverter base class and .

b) *SelectionRangeConverter*

Example Syntax:

c) *ToString*

[C#]	public	SelectionRangeConverter();
[C++]	public:	SelectionRangeConverter();

1 [VB] Public Sub New()

2 [JScript] public function SelectionRangeConverter();

3 *d) CanConvertFrom*

4
5 [C#] public override bool CanConvertFrom(ITypeDescriptorContext context,
6 Type sourceType);

7 [C++] public: bool CanConvertFrom(ITypeDescriptorContext* context, Type*
8 sourceType);

9 [VB] Overrides Public Function CanConvertFrom(ByVal context As
10 ITypeDescriptorContext, ByVal sourceType As Type) As Boolean

11 [JScript] public override function CanConvertFrom(context :
12 ITypeDescriptorContext, sourceType : Type) : Boolean;

13
14 *Description*

15 Determines if this converter can convert an object in the given source type to the
16 native type of the converter.

Return Value: True if this object can perform the conversion.

17 *e) CanConvertTo*

18
19 [C#] public override bool CanConvertTo(ITypeDescriptorContext context, Type
20 destinationType);

21 [C++] public: bool CanConvertTo(ITypeDescriptorContext* context, Type*
22 destinationType);

23 [VB] Overrides Public Function CanConvertTo(ByVal context As
24 ITypeDescriptorContext, ByVal destinationType As Type) As Boolean

```

1 [JScript] public override function CanConvertTo(context :
2 ITypeDescriptorContext, destinationType : Type) : Boolean;
3

```

4 *Description*

5 Gets a value indicating whether this converter can convert an object to the given destination type using the context.

6 *Return Value:* **true** if this converter can perform the conversion; otherwise, **false** .

7 The *context* parameter can be used to extract additional information about the environment this converter is being invoked from. This can be **null** , so always check. Also, properties on the context object can return **null** .

9 *f) ConvertFrom*

```

10
11
12 [C#] public override object ConvertFrom(ITypeDescriptorContext context,
13 CultureInfo culture, object value);

```

```

14 [C++] public: Object* ConvertFrom(ITypeDescriptorContext* context,
15 CultureInfo* culture, Object* value);

```

```

16 [VB] Overrides Public Function ConvertFrom(ByVal context As
17 ITypeDescriptorContext, ByVal culture As CultureInfo, ByVal value As Object)
18 As Object

```

```

19 [JScript] public override function ConvertFrom(context : ITypeDescriptorContext,
20 culture : CultureInfo, value : Object) : Object;
21

```

22 *Description*

23 Converts the given object to the converter's native type.

24 *Return Value:* The converted object. This will throw an exception if the conversion could not be performed.

g) *ConvertTo*

```
[C#] public override object ConvertTo(ITypeDescriptorContext context,
CultureInfo culture, object value, Type destinationType);
[C++] public: Object* ConvertTo(ITypeDescriptorContext* context, CultureInfo*
culture, Object* value, Type* destinationType);
[VB] Overrides Public Function ConvertTo(ByVal context As
ITypeDescriptorContext, ByVal culture As CultureInfo, ByVal value As Object,
ByVal destinationType As Type) As Object
[JScript] public override function ConvertTo(context : ITypeDescriptorContext,
culture : CultureInfo, value : Object, destinationType : Type) : Object;
```

Description

Converts the given object to another type. The most common types to convert are to and from a string object. The default implementation will make a call to ToString on the object if the object is valid and if the destination type is string. If this cannot convert to the destination type, this will throw a NotSupportedException.

Return Value: The converted object.

h) *CreateInstance*

```
[C#] public override object CreateInstance(ITypeDescriptorContext context,
IDictionary propertyValues);
[C++] public: Object* CreateInstance(ITypeDescriptorContext* context,
IDictionary* propertyValues);
[VB] Overrides Public Function CreateInstance(ByVal context As
ITypeDescriptorContext, ByVal propertyValues As IDictionary) As Object
```

```

1 [JScript]      public      override      function      CreateInstance(context      :
2 ITypeDescriptorContext,      propertyValues      :      IDictionary)      :      Object;

```

4 *Description*

5 Creates an instance of this type given a set of property values for the object.
 6 This is useful for objects that are immutable, but still want to provide changable
 7 properties.

Return Value: The newly created object, or null if the object could not be
 7 created. The default implementation returns null.

8 *i) GetCreateInstanceSupported*

```

10 [C#] public override bool GetCreateInstanceSupported(ITypeDescriptorContext
11 context);

```

```

12 [C++] public: bool GetCreateInstanceSupported(ITypeDescriptorContext*
13 context);

```

```

14 [VB] Overrides Public Function GetCreateInstanceSupported(ByVal context As
15 ITypeDescriptorContext) As Boolean

```

```

16 [JScript] public override function GetCreateInstanceSupported(context :
17 ITypeDescriptorContext) : Boolean;

```

19 *Description*

20 Determines if changing a value on this object should require a call to
 21 CreateInstance to create a new value.

Return Value: Returns true if CreateInstance should be called when a change is
 22 made to one or more properties of this object.

23 *j) GetProperties*

```

25 [C#]      public      override      PropertyDescriptorCollection

```

1 GetProperties(ITypeDescriptorContext context, object value, Attribute[]
2 attributes);

3 [C++] public: PropertyDescriptorCollection*
4 GetProperties(ITypeDescriptorContext* context, Object* value, Attribute*
5 attributes[]);

6 [VB] Overrides Public Function GetProperties(ByVal context As
7 ITypeDescriptorContext, ByVal value As Object, ByVal attributes() As Attribute)
8 As PropertyDescriptorCollection

9 [JScript] public override function GetProperties(context : ITypeDescriptorContext,
10 value : Object, attributes : Attribute[]) : PropertyDescriptorCollection;

12 *Description*

13 Retrieves the set of properties for this type. By default, a type has does not
14 return any properties. An easy implementation of this method can just call
15 TypeDescriptor.GetProperties for the correct data type.

16 *Return Value:* The set of properties that should be exposed for this data type. If
17 no properties should be exposed, thsi may return null. The default
18 implementation always returns null.

17 *k) GetPropertiesSupported*

19 [C#] public override bool GetPropertiesSupported(ITypeDescriptorContext
20 context);

21 [C++] public: bool GetPropertiesSupported(ITypeDescriptorContext* context);

22 [VB] Overrides Public Function GetPropertiesSupported(ByVal context As
23 ITypeDescriptorContext) As Boolean

24 [JScript] public override function GetPropertiesSupported(context :
25 ITypeDescriptorContext) : Boolean;

1
2 *Description*

3 Determines if this object supports properties. By default, this is false.

4 *Return Value:* Returns true if GetProperties should be called to find the properties of this object.

5 SendKeys class (System.Windows.Forms)

6 *a) ToString*

7
8
9 *Description*

10 Provides methods for sending keystrokes to an application.

11 Use **System.Windows.Forms.SendKeys** to send keystrokes and keystroke combinations to the active application. This class cannot be instantiated. To send a keystroke to a class and immediately continue with the flow of your program, use **System.Windows.Forms.SendKeys.Send(System.String)** . To wait for any processes started by the keystroke, use **System.Windows.Forms.SendKeys.SendWait(System.String)** .

12
13
14
15 *b) Flush*

16
17 [C#] public static void Flush();
18 [C++] public: static void Flush();
19 [VB] Public Shared Sub Flush()
20 [JScript] public static function Flush();

21
22 *Description*

23 Processes all the Windows messages currently in the message queue.

24 Because there is no managed method to activate another application, you can either use this class within the current application or use native Windows

25

methods, such as **FindWindow** and **SetForegroundWindow** , to force focus on other applications.

c) *Send*

```
[C#]      public      static      void      Send(string      keys);
[C++]     public:      static      void      Send(String*      keys);
[VB]      Public      Shared      Sub      Send(ByVal      keys      As      String)
[JScript] public      static      function Send(keys      :      String);
```

Description

Sends keystrokes to the active application.

Each key is represented by one or more characters. To specify a single keyboard character, use the character itself. For example, to represent the letter A, pass in the string "A" to the method. To represent more than one character, append each additional character to the one preceding it. To represent the letters A, B, and C, specify the parameter as "ABC". The string of keystrokes to send.

d) *SendWait*

```
[C#]      public      static      void      SendWait(string      keys);
[C++]     public:      static      void      SendWait(String*      keys);
[VB]      Public      Shared      Sub      SendWait(ByVal      keys      As      String)
[JScript] public      static      function SendWait(keys      :      String);
```

Description

Sends the given keys to the active application, and then waits for the messages to be processed.

Use **System.Windows.Forms.SendKeys.SendWait(System.String)** to send keystrokes or combinations of keystrokes to the active application and wait for the keystroke messages to be processed. You can use this method to send

1 keystrokes to an application and wait for any processes that are started by the
2 keystrokes to be completed. This can be important if the other application must
3 finish before your application can continue. The string of keystrokes to send.

4 Shortcut enumeration (System.Windows.Forms)

5 *a) ToString*

6
7 *Description*

8 Specifies shortcut keys that can be used by menu items.

9 Use the members of this enumeration to set the value of the
10 **System.Windows.Forms.MenuItem.Shortcut** property of the
11 **System.Windows.Forms.MenuItem** class.

12 *b) ToString*

13 [C#]	public	const	Shortcut	Alt0;
14 [C++]	public:	const	Shortcut	Alt0;
15 [VB]	Public	Const	Alt0 As	Shortcut
16 [JScript]	public	var	Alt0 :	Shortcut;

17
18 *Description*

19 The shortcut keys ALT+0.

20 *c) ToString*

22 [C#]	public	const	Shortcut	Alt1;
23 [C++]	public:	const	Shortcut	Alt1;
24 [VB]	Public	Const	Alt1 As	Shortcut
25 [JScript]	public	var	Alt1 :	Shortcut;

Description

The shortcut keys ALT+1.

d) ToString

[C#]	public	const	Shortcut	Alt2;	
[C++]	public:	const	Shortcut	Alt2;	
[VB]	Public	Const	Alt2	As	Shortcut
[JScript]	public	var	Alt2	:	Shortcut;

Description

The shortcut keys ALT+2.

e) ToString

[C#]	public	const	Shortcut	Alt3;	
[C++]	public:	const	Shortcut	Alt3;	
[VB]	Public	Const	Alt3	As	Shortcut
[JScript]	public	var	Alt3	:	Shortcut;

Description

The shortcut keys ALT+3.

f) ToString

[C#]	public	const	Shortcut	Alt4;
[C++]	public:	const	Shortcut	Alt4;

[VB]	Public	Const	Alt4	As	Shortcut
[JScript]	public	var	Alt4	:	Shortcut;

Description

The shortcut keys ALT+4.

g) ToString

[C#]	public	const		Shortcut	Alt5;
[C++]	public:	const		Shortcut	Alt5;
[VB]	Public	Const	Alt5	As	Shortcut
[JScript]	public	var	Alt5	:	Shortcut;

Description

The shortcut keys ALT+5.

h) ToString

[C#]	public	const		Shortcut	Alt6;
[C++]	public:	const		Shortcut	Alt6;
[VB]	Public	Const	Alt6	As	Shortcut
[JScript]	public	var	Alt6	:	Shortcut;

Description

The shortcut keys ALT+6.

i) ToString

[C#]	public	const	Shortcut	Alt7;	
[C++]	public:	const	Shortcut	Alt7;	
[VB]	Public	Const	Alt7	As	Shortcut
[JScript]	public	var	Alt7	:	Shortcut;

Description

The shortcut keys ALT+7.

j) ToString

[C#]	public	const	Shortcut	Alt8;	
[C++]	public:	const	Shortcut	Alt8;	
[VB]	Public	Const	Alt8	As	Shortcut
[JScript]	public	var	Alt8	:	Shortcut;

Description

The shortcut keys ALT+8.

k) ToString

[C#]	public	const	Shortcut	Alt9;	
[C++]	public:	const	Shortcut	Alt9;	
[VB]	Public	Const	Alt9	As	Shortcut
[JScript]	public	var	Alt9	:	Shortcut;

Description

The shortcut keys ALT+9.

l) ToString

[C#]	public	const	Shortcut	AltBksp;
[C++]	public:	const	Shortcut	AltBksp;
[VB]	Public	Const	AltBksp	As Shortcut
[JScript]	public	var	AltBksp	: Shortcut;

Description

The shortcut keys ALT+BACKSPACE.

m) ToString

[C#]	public	const	Shortcut	AltF1;
[C++]	public:	const	Shortcut	AltF1;
[VB]	Public	Const	AltF1	As Shortcut
[JScript]	public	var	AltF1	: Shortcut;

Description

The shortcut keys ALT+F1.

n) ToString

[C#]	public	const	Shortcut	AltF10;
[C++]	public:	const	Shortcut	AltF10;

[VB]	Public	Const	AltF10	As	Shortcut
[JScript]	public	var	AltF10	:	Shortcut;

Description

The shortcut keys ALT+F10.

o) ToString

[C#]	public	const	Shortcut	AltF11;	
[C++]	public:	const	Shortcut	AltF11;	
[VB]	Public	Const	AltF11	As	Shortcut
[JScript]	public	var	AltF11	:	Shortcut;

Description

The shortcut keys ALT+F11.

p) ToString

[C#]	public	const	Shortcut	AltF12;	
[C++]	public:	const	Shortcut	AltF12;	
[VB]	Public	Const	AltF12	As	Shortcut
[JScript]	public	var	AltF12	:	Shortcut;

Description

The shortcut keys ALT+F12.

q) ToString

[C#]	public	const	Shortcut	AltF2;
[C++]	public:	const	Shortcut	AltF2;
[VB]	Public	Const	AltF2	As Shortcut
[JScript]	public	var	AltF2	: Shortcut;

Description

The shortcut keys ALT+F2.

r) ToString

[C#]	public	const	Shortcut	AltF3;
[C++]	public:	const	Shortcut	AltF3;
[VB]	Public	Const	AltF3	As Shortcut
[JScript]	public	var	AltF3	: Shortcut;

Description

The shortcut keys ALT+F3.

s) ToString

[C#]	public	const	Shortcut	AltF4;
[C++]	public:	const	Shortcut	AltF4;
[VB]	Public	Const	AltF4	As Shortcut
[JScript]	public	var	AltF4	: Shortcut;

Description

The shortcut keys ALT+F4.

t) ToString

[C#]	public	const	Shortcut	AltF5;	
[C++]	public:	const	Shortcut	AltF5;	
[VB]	Public	Const	AltF5	As	Shortcut
[JScript]	public	var	AltF5	:	Shortcut;

Description

The shortcut keys ALT+F5.

u) ToString

[C#]	public	const	Shortcut	AltF6;	
[C++]	public:	const	Shortcut	AltF6;	
[VB]	Public	Const	AltF6	As	Shortcut
[JScript]	public	var	AltF6	:	Shortcut;

Description

The shortcut keys ALT+F6.

v) ToString

[C#]	public	const	Shortcut	AltF7;
[C++]	public:	const	Shortcut	AltF7;

[VB]	Public	Const	AltF7	As	Shortcut
[JScript]	public	var	AltF7	:	Shortcut;

Description

The shortcut keys ALT+F7.

w) ToString

[C#]	public	const	Shortcut	AltF8;	
[C++]	public:	const	Shortcut	AltF8;	
[VB]	Public	Const	AltF8	As	Shortcut
[JScript]	public	var	AltF8	:	Shortcut;

Description

The shortcut keys ALT+F8.

x) ToString

[C#]	public	const	Shortcut	AltF9;	
[C++]	public:	const	Shortcut	AltF9;	
[VB]	Public	Const	AltF9	As	Shortcut
[JScript]	public	var	AltF9	:	Shortcut;

Description

The shortcut keys ALT+F9.

y) *ToString*

[C#]	public	const	Shortcut	Ctrl0;
[C++]	public:	const	Shortcut	Ctrl0;
[VB]	Public	Const	Ctrl0	As Shortcut
[JScript]	public	var	Ctrl0	: Shortcut;

Description

The shortcut keys CTRL+0.

z) *ToString*

[C#]	public	const	Shortcut	Ctrl1;
[C++]	public:	const	Shortcut	Ctrl1;
[VB]	Public	Const	Ctrl1	As Shortcut
[JScript]	public	var	Ctrl1	: Shortcut;

Description

The shortcut keys CTRL+1.

aa) *ToString*

[C#]	public	const	Shortcut	Ctrl2;
[C++]	public:	const	Shortcut	Ctrl2;
[VB]	Public	Const	Ctrl2	As Shortcut
[JScript]	public	var	Ctrl2	: Shortcut;

Description

The shortcut keys CTRL+2.

bb) ToString

[C#]	public	const	Shortcut	Ctrl3;	
[C++]	public:	const	Shortcut	Ctrl3;	
[VB]	Public	Const	Ctrl3	As	Shortcut
[JScript]	public	var	Ctrl3	:	Shortcut;

Description

The shortcut keys CTRL+3.

cc) ToString

[C#]	public	const	Shortcut	Ctrl4;	
[C++]	public:	const	Shortcut	Ctrl4;	
[VB]	Public	Const	Ctrl4	As	Shortcut
[JScript]	public	var	Ctrl4	:	Shortcut;

Description

The shortcut keys CTRL+4.

dd) ToString

[C#]	public	const	Shortcut	Ctrl5;
[C++]	public:	const	Shortcut	Ctrl5;

[VB]	Public	Const	Ctrl5	As	Shortcut
[JScript]	public	var	Ctrl5	:	Shortcut;

Description

The shortcut keys CTRL+5.

ee) ToString

[C#]	public	const	Shortcut	Ctrl6;	
[C++]	public:	const	Shortcut	Ctrl6;	
[VB]	Public	Const	Ctrl6	As	Shortcut
[JScript]	public	var	Ctrl6	:	Shortcut;

Description

The shortcut keys CTRL+6.

ff) ToString

[C#]	public	const	Shortcut	Ctrl7;	
[C++]	public:	const	Shortcut	Ctrl7;	
[VB]	Public	Const	Ctrl7	As	Shortcut
[JScript]	public	var	Ctrl7	:	Shortcut;

Description

The shortcut keys CTRL+7.

gg) ToString

[C#]	public	const	Shortcut	Ctrl8;
[C++]	public:	const	Shortcut	Ctrl8;
[VB]	Public	Const	Ctrl8 As	Shortcut
[JScript]	public	var	Ctrl8 :	Shortcut;

Description

The shortcut keys CTRL+8.

hh) ToString

[C#]	public	const	Shortcut	Ctrl9;
[C++]	public:	const	Shortcut	Ctrl9;
[VB]	Public	Const	Ctrl9 As	Shortcut
[JScript]	public	var	Ctrl9 :	Shortcut;

Description

The shortcut keys CTRL+9.

ii) ToString

[C#]	public	const	Shortcut	CtrlA;
[C++]	public:	const	Shortcut	CtrlA;
[VB]	Public	Const	CtrlA As	Shortcut
[JScript]	public	var	CtrlA :	Shortcut;

Description

The shortcut keys CTRL+A.

jj) ToString

[C#]	public	const	Shortcut	CtrlB;
[C++]	public:	const	Shortcut	CtrlB;
[VB]	Public	Const	CtrlB As	Shortcut
[JScript]	public	var	CtrlB :	Shortcut;

Description

The shortcut keys CTRL+B.

kk) ToString

[C#]	public	const	Shortcut	CtrlC;
[C++]	public:	const	Shortcut	CtrlC;
[VB]	Public	Const	CtrlC As	Shortcut
[JScript]	public	var	CtrlC :	Shortcut;

Description

The shortcut keys CTRL+C.

ll) ToString

[C#]	public	const	Shortcut	CtrlD;
[C++]	public:	const	Shortcut	CtrlD;

[VB]	Public	Const	CtrlD	As	Shortcut
[JScript]	public	var	CtrlD	:	Shortcut;

Description

The shortcut keys CTRL+D.

mm) ToString

[C#]	public	const	Shortcut		CtrlDel;
[C++]	public:	const	Shortcut		CtrlDel;
[VB]	Public	Const	CtrlDel	As	Shortcut
[JScript]	public	var	CtrlDel	:	Shortcut;

Description

The shortcut keys CTRL+DELETE.

nn) ToString

[C#]	public	const	Shortcut		CtrlE;
[C++]	public:	const	Shortcut		CtrlE;
[VB]	Public	Const	CtrlE	As	Shortcut
[JScript]	public	var	CtrlE	:	Shortcut;

Description

The shortcut keys CTRL+E.

oo) ToString

[C#]	public	const	Shortcut	CtrlF;	
[C++]	public:	const	Shortcut	CtrlF;	
[VB]	Public	Const	CtrlF	As	Shortcut
[JScript]	public	var	CtrlF	:	Shortcut;

Description

The shortcut keys CTRL+F.

pp) ToString

[C#]	public	const	Shortcut	CtrlF1;	
[C++]	public:	const	Shortcut	CtrlF1;	
[VB]	Public	Const	CtrlF1	As	Shortcut
[JScript]	public	var	CtrlF1	:	Shortcut;

Description

The shortcut keys CTRL+F1.

qq) ToString

[C#]	public	const	Shortcut	CtrlF10;	
[C++]	public:	const	Shortcut	CtrlF10;	
[VB]	Public	Const	CtrlF10	As	Shortcut
[JScript]	public	var	CtrlF10	:	Shortcut;

Description

The shortcut keys CTRL+F10.

rr) ToString

[C#]	public	const	Shortcut	CtrlF11;
[C++]	public:	const	Shortcut	CtrlF11;
[VB]	Public	Const	CtrlF11	As Shortcut
[JScript]	public	var	CtrlF11	: Shortcut;

Description

The shortcut keys CTRL+F11.

ss) ToString

[C#]	public	const	Shortcut	CtrlF12;
[C++]	public:	const	Shortcut	CtrlF12;
[VB]	Public	Const	CtrlF12	As Shortcut
[JScript]	public	var	CtrlF12	: Shortcut;

Description

The shortcut keys CTRL+F12.

tt) ToString

[C#]	public	const	Shortcut	CtrlF2;
[C++]	public:	const	Shortcut	CtrlF2;

[VB]	Public	Const	CtrlF2	As	Shortcut
[JScript]	public	var	CtrlF2	:	Shortcut;

Description

The shortcut keys CTRL+F2.

uu) ToString

[C#]	public	const	Shortcut	CtrlF3;	
[C++]	public:	const	Shortcut	CtrlF3;	
[VB]	Public	Const	CtrlF3	As	Shortcut
[JScript]	public	var	CtrlF3	:	Shortcut;

Description

The shortcut keys CTRL+F3.

vv) ToString

[C#]	public	const	Shortcut	CtrlF4;	
[C++]	public:	const	Shortcut	CtrlF4;	
[VB]	Public	Const	CtrlF4	As	Shortcut
[JScript]	public	var	CtrlF4	:	Shortcut;

Description

The shortcut keys CTRL+F4.

ww) ToString

[C#]	public	const	Shortcut	CtrlF5;
[C++]	public:	const	Shortcut	CtrlF5;
[VB]	Public	Const	CtrlF5	As Shortcut
[JScript]	public	var	CtrlF5	: Shortcut;

Description

The shortcut keys CTRL+F5.

xx) ToString

[C#]	public	const	Shortcut	CtrlF6;
[C++]	public:	const	Shortcut	CtrlF6;
[VB]	Public	Const	CtrlF6	As Shortcut
[JScript]	public	var	CtrlF6	: Shortcut;

Description

The shortcut keys CTRL+F6.

yy) ToString

[C#]	public	const	Shortcut	CtrlF7;
[C++]	public:	const	Shortcut	CtrlF7;
[VB]	Public	Const	CtrlF7	As Shortcut
[JScript]	public	var	CtrlF7	: Shortcut;

Description

The shortcut keys CTRL+F7.

zz) ToString

[C#]	public	const	Shortcut	CtrlF8;
[C++]	public:	const	Shortcut	CtrlF8;
[VB]	Public	Const	CtrlF8	As Shortcut
[JScript]	public	var	CtrlF8	: Shortcut;

Description

The shortcut keys CTRL+F8.

aaa) ToString

[C#]	public	const	Shortcut	CtrlF9;
[C++]	public:	const	Shortcut	CtrlF9;
[VB]	Public	Const	CtrlF9	As Shortcut
[JScript]	public	var	CtrlF9	: Shortcut;

Description

The shortcut keys CTRL+F9.

bbb) ToString

[C#]	public	const	Shortcut	CtrlG;
[C++]	public:	const	Shortcut	CtrlG;

[VB]	Public	Const	CtrlG	As	Shortcut
[JScript]	public	var	CtrlG	:	Shortcut;

Description

The shortcut keys CTRL+G.

ccc) ToString

[C#]	public	const	Shortcut	CtrlH;	
[C++]	public:	const	Shortcut	CtrlH;	
[VB]	Public	Const	CtrlH	As	Shortcut
[JScript]	public	var	CtrlH	:	Shortcut;

Description

The shortcut keys CTRL+H.

ddd) ToString

[C#]	public	const	Shortcut	CtrlI;	
[C++]	public:	const	Shortcut	CtrlI;	
[VB]	Public	Const	CtrlI	As	Shortcut
[JScript]	public	var	CtrlI	:	Shortcut;

Description

The shortcut keys CTRL+I.

eee) ToString

[C#]	public	const	Shortcut	CtrlIns;
[C++]	public:	const	Shortcut	CtrlIns;
[VB]	Public	Const	CtrlIns	As Shortcut
[JScript]	public	var	CtrlIns	: Shortcut;

Description

The shortcut keys CTRL+INSERT.

fff) ToString

[C#]	public	const	Shortcut	CtrlJ;
[C++]	public:	const	Shortcut	CtrlJ;
[VB]	Public	Const	CtrlJ	As Shortcut
[JScript]	public	var	CtrlJ	: Shortcut;

Description

The shortcut keys CTRL+J.

ggg) ToString

[C#]	public	const	Shortcut	CtrlK;
[C++]	public:	const	Shortcut	CtrlK;
[VB]	Public	Const	CtrlK	As Shortcut
[JScript]	public	var	CtrlK	: Shortcut;

Description

The shortcut keys CTRL+K.

hhh) ToString

[C#]	public	const	Shortcut	CtrlL;	
[C++]	public:	const	Shortcut	CtrlL;	
[VB]	Public	Const	CtrlL	As	Shortcut
[JScript]	public	var	CtrlL	:	Shortcut;

Description

The shortcut keys CTRL+L.

iii) ToString

[C#]	public	const	Shortcut	CtrlM;	
[C++]	public:	const	Shortcut	CtrlM;	
[VB]	Public	Const	CtrlM	As	Shortcut
[JScript]	public	var	CtrlM	:	Shortcut;

Description

The shortcut keys CTRL+M.

jjj) ToString

[C#]	public	const	Shortcut	CtrlN;
[C++]	public:	const	Shortcut	CtrlN;

1	[VB]	Public	Const	CtrlN	As	Shortcut
2	[JScript]	public	var	CtrlN	:	Shortcut;

3

4 *Description*

5 The shortcut keys CTRL+N.

6 *kkk) ToString*

7

8	[C#]	public	const	Shortcut		CtrlO;
9	[C++]	public:	const	Shortcut		CtrlO;
10	[VB]	Public	Const	CtrlO	As	Shortcut
11	[JScript]	public	var	CtrlO	:	Shortcut;

12

13 *Description*

14 The shortcut keys CTRL+O.

15 *III) ToString*

16

17	[C#]	public	const	Shortcut		CtrlP;
18	[C++]	public:	const	Shortcut		CtrlP;
19	[VB]	Public	Const	CtrlP	As	Shortcut
20	[JScript]	public	var	CtrlP	:	Shortcut;

21

22 *Description*

23 The shortcut keys CTRL+P.

24

25

mmm) ToString

[C#]	public	const	Shortcut	CtrlQ;	
[C++]	public:	const	Shortcut	CtrlQ;	
[VB]	Public	Const	CtrlQ	As	Shortcut
[JScript]	public	var	CtrlQ	:	Shortcut;

Description

The shortcut keys CTRL+Q.

nnn) ToString

[C#]	public	const	Shortcut	CtrlR;	
[C++]	public:	const	Shortcut	CtrlR;	
[VB]	Public	Const	CtrlR	As	Shortcut
[JScript]	public	var	CtrlR	:	Shortcut;

Description

The shortcut keys CTRL+R.

ooo) ToString

[C#]	public	const	Shortcut	CtrlS;	
[C++]	public:	const	Shortcut	CtrlS;	
[VB]	Public	Const	CtrlS	As	Shortcut
[JScript]	public	var	CtrlS	:	Shortcut;

Description

The shortcut keys CTRL+S.

ppp) ToString

[C#]	public	const	Shortcut	CtrlShift0;
[C++]	public:	const	Shortcut	CtrlShift0;
[VB]	Public	Const	CtrlShift0	As Shortcut
[JScript]	public	var	CtrlShift0	: Shortcut;

Description

The shortcut keys CTRL+SHIFT+0.

qqq) ToString

[C#]	public	const	Shortcut	CtrlShift1;
[C++]	public:	const	Shortcut	CtrlShift1;
[VB]	Public	Const	CtrlShift1	As Shortcut
[JScript]	public	var	CtrlShift1	: Shortcut;

Description

The shortcut keys CTRL+SHIFT+1.

rrr) ToString

[C#]	public	const	Shortcut	CtrlShift2;
[C++]	public:	const	Shortcut	CtrlShift2;

[VB]	Public	Const	CtrlShift2	As	Shortcut
[JScript]	public	var	CtrlShift2	:	Shortcut;

Description

The shortcut keys CTRL+SHIFT+2.

sss) ToString

[C#]	public	const	Shortcut	CtrlShift3;	
[C++]	public:	const	Shortcut	CtrlShift3;	
[VB]	Public	Const	CtrlShift3	As	Shortcut
[JScript]	public	var	CtrlShift3	:	Shortcut;

Description

The shortcut keys CTRL+SHIFT+3.

ttt) ToString

[C#]	public	const	Shortcut	CtrlShift4;	
[C++]	public:	const	Shortcut	CtrlShift4;	
[VB]	Public	Const	CtrlShift4	As	Shortcut
[JScript]	public	var	CtrlShift4	:	Shortcut;

Description

The shortcut keys CTRL+SHIFT+4.

uuu) ToString

[C#]	public	const	Shortcut	CtrlShift5;
[C++]	public:	const	Shortcut	CtrlShift5;
[VB]	Public	Const	CtrlShift5	As Shortcut
[JScript]	public	var	CtrlShift5	: Shortcut;

Description

The shortcut keys CTRL+SHIFT+5.

vvv) ToString

[C#]	public	const	Shortcut	CtrlShift6;
[C++]	public:	const	Shortcut	CtrlShift6;
[VB]	Public	Const	CtrlShift6	As Shortcut
[JScript]	public	var	CtrlShift6	: Shortcut;

Description

The shortcut keys CTRL+SHIFT+6.

www) ToString

[C#]	public	const	Shortcut	CtrlShift7;
[C++]	public:	const	Shortcut	CtrlShift7;
[VB]	Public	Const	CtrlShift7	As Shortcut
[JScript]	public	var	CtrlShift7	: Shortcut;

Description

The shortcut keys CTRL+SHIFT+7.

xxx) ToString

[C#]	public	const	Shortcut	CtrlShift8;
[C++]	public:	const	Shortcut	CtrlShift8;
[VB]	Public	Const	CtrlShift8	As Shortcut
[JScript]	public	var	CtrlShift8	: Shortcut;

Description

The shortcut keys CTRL+SHIFT+8.

yyy) ToString

[C#]	public	const	Shortcut	CtrlShift9;
[C++]	public:	const	Shortcut	CtrlShift9;
[VB]	Public	Const	CtrlShift9	As Shortcut
[JScript]	public	var	CtrlShift9	: Shortcut;

Description

The shortcut keys CTRL+SHIFT+9.

zzz) ToString

[C#]	public	const	Shortcut	CtrlShiftA;
[C++]	public:	const	Shortcut	CtrlShiftA;

[VB]	Public	Const	CtrlShiftA	As	Shortcut
[JScript]	public	var	CtrlShiftA	:	Shortcut;

Description

The shortcut keys CTRL+SHIFT+A.

aaaa) ToString

[C#]	public	const	Shortcut		CtrlShiftB;
[C++]	public:	const	Shortcut		CtrlShiftB;
[VB]	Public	Const	CtrlShiftB	As	Shortcut
[JScript]	public	var	CtrlShiftB	:	Shortcut;

Description

The shortcut keys CTRL+SHIFT+B.

bbbb) ToString

[C#]	public	const	Shortcut		CtrlShiftC;
[C++]	public:	const	Shortcut		CtrlShiftC;
[VB]	Public	Const	CtrlShiftC	As	Shortcut
[JScript]	public	var	CtrlShiftC	:	Shortcut;

Description

The shortcut keys CTRL+SHIFT+C.

cccc) ToString

[C#]	public	const	Shortcut	CtrlShiftD;
[C++]	public:	const	Shortcut	CtrlShiftD;
[VB]	Public	Const	CtrlShiftD	As Shortcut
[JScript]	public	var	CtrlShiftD	: Shortcut;

Description

The shortcut keys CTRL+SHIFT+D.

dddd) ToString

[C#]	public	const	Shortcut	CtrlShiftE;
[C++]	public:	const	Shortcut	CtrlShiftE;
[VB]	Public	Const	CtrlShiftE	As Shortcut
[JScript]	public	var	CtrlShiftE	: Shortcut;

Description

The shortcut keys CTRL+SHIFT+E.

eeee) ToString

[C#]	public	const	Shortcut	CtrlShiftF;
[C++]	public:	const	Shortcut	CtrlShiftF;
[VB]	Public	Const	CtrlShiftF	As Shortcut
[JScript]	public	var	CtrlShiftF	: Shortcut;

Description

The shortcut keys CTRL+SHIFT+F.

ffff) ToString

[C#]	public	const	Shortcut	CtrlShiftF1;
[C++]	public:	const	Shortcut	CtrlShiftF1;
[VB]	Public	Const	CtrlShiftF1	As Shortcut
[JScript]	public	var	CtrlShiftF1	: Shortcut;

Description

The shortcut keys CTRL+SHIFT+F1.

gggg) ToString

[C#]	public	const	Shortcut	CtrlShiftF10;
[C++]	public:	const	Shortcut	CtrlShiftF10;
[VB]	Public	Const	CtrlShiftF10	As Shortcut
[JScript]	public	var	CtrlShiftF10	: Shortcut;

Description

The shortcut keys CTRL+SHIFT+F10.

hhhh) ToString

[C#]	public	const	Shortcut	CtrlShiftF11;
[C++]	public:	const	Shortcut	CtrlShiftF11;

[VB]	Public	Const	CtrlShiftF11	As	Shortcut
[JScript]	public	var	CtrlShiftF11	:	Shortcut;

Description

The shortcut keys CTRL+SHIFT+F11.

iii) ToString

[C#]	public	const	Shortcut		CtrlShiftF12;
[C++]	public:	const	Shortcut		CtrlShiftF12;
[VB]	Public	Const	CtrlShiftF12	As	Shortcut
[JScript]	public	var	CtrlShiftF12	:	Shortcut;

Description

The shortcut keys CTRL+SHIFT+F12.

iiii) ToString

[C#]	public	const	Shortcut		CtrlShiftF2;
[C++]	public:	const	Shortcut		CtrlShiftF2;
[VB]	Public	Const	CtrlShiftF2	As	Shortcut
[JScript]	public	var	CtrlShiftF2	:	Shortcut;

Description

The shortcut keys CTRL+SHIFT+F2.

kkkk) ToString

[C#]	public	const	Shortcut	CtrlShiftF3;
[C++]	public:	const	Shortcut	CtrlShiftF3;
[VB]	Public	Const	CtrlShiftF3	As Shortcut
[JScript]	public	var	CtrlShiftF3	: Shortcut;

Description

The shortcut keys CTRL+SHIFT+F3.

llll) ToString

[C#]	public	const	Shortcut	CtrlShiftF4;
[C++]	public:	const	Shortcut	CtrlShiftF4;
[VB]	Public	Const	CtrlShiftF4	As Shortcut
[JScript]	public	var	CtrlShiftF4	: Shortcut;

Description

The shortcut keys CTRL+SHIFT+F4.

mmmm) ToString

[C#]	public	const	Shortcut	CtrlShiftF5;
[C++]	public:	const	Shortcut	CtrlShiftF5;
[VB]	Public	Const	CtrlShiftF5	As Shortcut
[JScript]	public	var	CtrlShiftF5	: Shortcut;

Description

The shortcut keys CTRL+SHIFT+F5.

nnnn) ToString

[C#]	public	const	Shortcut		CtrlShiftF6;
[C++]	public:	const	Shortcut		CtrlShiftF6;
[VB]	Public	Const	CtrlShiftF6	As	Shortcut
[JScript]	public	var	CtrlShiftF6	:	Shortcut;

Description

The shortcut keys CTRL+SHIFT+F6.

oooo) ToString

[C#]	public	const	Shortcut		CtrlShiftF7;
[C++]	public:	const	Shortcut		CtrlShiftF7;
[VB]	Public	Const	CtrlShiftF7	As	Shortcut
[JScript]	public	var	CtrlShiftF7	:	Shortcut;

Description

The shortcut keys CTRL+SHIFT+F7.

pppp) ToString

[C#]	public	const	Shortcut		CtrlShiftF8;
[C++]	public:	const	Shortcut		CtrlShiftF8;

[VB]	Public	Const	CtrlShiftF8	As	Shortcut
[JScript]	public	var	CtrlShiftF8	:	Shortcut;

Description

The shortcut keys CTRL+SHIFT+F8.

qqqq) ToString

[C#]	public	const	Shortcut		CtrlShiftF9;
[C++]	public:	const	Shortcut		CtrlShiftF9;
[VB]	Public	Const	CtrlShiftF9	As	Shortcut
[JScript]	public	var	CtrlShiftF9	:	Shortcut;

Description

The shortcut keys CTRL+SHIFT+F9.

rrrr) ToString

[C#]	public	const	Shortcut		CtrlShiftG;
[C++]	public:	const	Shortcut		CtrlShiftG;
[VB]	Public	Const	CtrlShiftG	As	Shortcut
[JScript]	public	var	CtrlShiftG	:	Shortcut;

Description

The shortcut keys CTRL+SHIFT+G.

ssss) ToString

[C#]	public	const	Shortcut	CtrlShiftH;
[C++]	public:	const	Shortcut	CtrlShiftH;
[VB]	Public	Const	CtrlShiftH	As Shortcut
[JScript]	public	var	CtrlShiftH	: Shortcut;

Description

The shortcut keys CTRL+SHIFT+H.

tttt) ToString

[C#]	public	const	Shortcut	CtrlShiftI;
[C++]	public:	const	Shortcut	CtrlShiftI;
[VB]	Public	Const	CtrlShiftI	As Shortcut
[JScript]	public	var	CtrlShiftI	: Shortcut;

Description

The shortcut keys CTRL+SHIFT+I.

uuuu) ToString

[C#]	public	const	Shortcut	CtrlShiftJ;
[C++]	public:	const	Shortcut	CtrlShiftJ;
[VB]	Public	Const	CtrlShiftJ	As Shortcut
[JScript]	public	var	CtrlShiftJ	: Shortcut;

Description

The shortcut keys CTRL+SHIFT+J.

vvvv) ToString

[C#]	public	const	Shortcut		CtrlShiftK;
[C++]	public:	const	Shortcut		CtrlShiftK;
[VB]	Public	Const	CtrlShiftK	As	Shortcut
[JScript]	public	var	CtrlShiftK	:	Shortcut;

Description

The shortcut keys CTRL+SHIFT+K.

www)ToString

[C#]	public	const	Shortcut		CtrlShiftL;
[C++]	public:	const	Shortcut		CtrlShiftL;
[VB]	Public	Const	CtrlShiftL	As	Shortcut
[JScript]	public	var	CtrlShiftL	:	Shortcut;

Description

The shortcut keys CTRL+SHIFT+L.

xxxx) ToString

[C#]	public	const	Shortcut		CtrlShiftM;
[C++]	public:	const	Shortcut		CtrlShiftM;

[VB]	Public	Const	CtrlShiftM	As	Shortcut
[JScript]	public	var	CtrlShiftM	:	Shortcut;

Description

The shortcut keys CTRL+SHIFT+M.

yyyy) ToString

[C#]	public	const	Shortcut		CtrlShiftN;
[C++]	public:	const	Shortcut		CtrlShiftN;
[VB]	Public	Const	CtrlShiftN	As	Shortcut
[JScript]	public	var	CtrlShiftN	:	Shortcut;

Description

The shortcut keys CTRL+SHIFT+N.

zzzz) ToString

[C#]	public	const	Shortcut		CtrlShiftO;
[C++]	public:	const	Shortcut		CtrlShiftO;
[VB]	Public	Const	CtrlShiftO	- As	Shortcut
[JScript]	public	var	CtrlShiftO	:	Shortcut;

Description

The shortcut keys CTRL+SHIFT+O.

aaaaa) ToString

[C#]	public	const	Shortcut		CtrlShiftP;
[C++]	public:	const	Shortcut		CtrlShiftP;
[VB]	Public	Const	CtrlShiftP	As	Shortcut
[JScript]	public	var	CtrlShiftP	:	Shortcut;

Description

The shortcut keys CTRL+SHIFT+P.

bbbbbb) ToString

[C#]	public	const	Shortcut		CtrlShiftQ;
[C++]	public:	const	Shortcut		CtrlShiftQ;
[VB]	Public	Const	CtrlShiftQ	As	Shortcut
[JScript]	public	var	CtrlShiftQ	:	Shortcut;

Description

The shortcut keys CTRL+SHIFT+Q.

cccccc) ToString

[C#]	public	const	Shortcut		CtrlShiftR;
[C++]	public:	const	Shortcut		CtrlShiftR;
[VB]	Public	Const	CtrlShiftR	As	Shortcut
[JScript]	public	var	CtrlShiftR	:	Shortcut;

Description

The shortcut keys CTRL+SHIFT+R.

dddd) ToString

[C#]	public	const	Shortcut	CtrlShiftS;
[C++]	public:	const	Shortcut	CtrlShiftS;
[VB]	Public	Const	CtrlShiftS	As Shortcut
[JScript]	public	var	CtrlShiftS	: Shortcut;

Description

The shortcut keys CTRL+SHIFT+S.

eeee) ToString

[C#]	public	const	Shortcut	CtrlShiftT;
[C++]	public:	const	Shortcut	CtrlShiftT;
[VB]	Public	Const	CtrlShiftT	As Shortcut
[JScript]	public	var	CtrlShiftT	: Shortcut;

Description

The shortcut keys CTRL+SHIFT+T.

ffff) ToString

[C#]	public	const	Shortcut	CtrlShiftU;
[C++]	public:	const	Shortcut	CtrlShiftU;

[VB]	Public	Const	CtrlShiftU	As	Shortcut
[JScript]	public	var	CtrlShiftU	:	Shortcut;

Description

The shortcut keys CTRL+SHIFT+U.

ggggg) ToString

[C#]	public	const	Shortcut		CtrlShiftV;
[C++]	public:	const	Shortcut		CtrlShiftV;
[VB]	Public	Const	CtrlShiftV	As	Shortcut
[JScript]	public	var	CtrlShiftV	:	Shortcut;

Description

The shortcut keys CTRL+SHIFT+V.

hhhhh) ToString

[C#]	public	const	Shortcut		CtrlShiftW;
[C++]	public:	const	Shortcut		CtrlShiftW;
[VB]	Public	Const	CtrlShiftW	As	Shortcut
[JScript]	public	var	CtrlShiftW	:	Shortcut;

Description

The shortcut keys CTRL+SHIFT+W.

iiii) ToString

[C#]	public	const	Shortcut	CtrlShiftX;
[C++]	public:	const	Shortcut	CtrlShiftX;
[VB]	Public	Const	CtrlShiftX	As Shortcut
[JScript]	public	var	CtrlShiftX	: Shortcut;

Description

The shortcut keys CTRL+SHIFT+X.

jjjj) ToString

[C#]	public	const	Shortcut	CtrlShiftY;
[C++]	public:	const	Shortcut	CtrlShiftY;
[VB]	Public	Const	CtrlShiftY	As Shortcut
[JScript]	public	var	CtrlShiftY	: Shortcut;

Description

The shortcut keys CTRL+SHIFT+Y.

kkkkk) ToString

[C#]	public	const	Shortcut	CtrlShiftZ;
[C++]	public:	const	Shortcut	CtrlShiftZ;
[VB]	Public	Const	CtrlShiftZ	As Shortcut
[JScript]	public	var	CtrlShiftZ	: Shortcut;

Description

The shortcut keys CTRL+SHIFT+Z.

lllll) ToString

[C#]	public	const	Shortcut	CtrlT;	
[C++]	public:	const	Shortcut	CtrlT;	
[VB]	Public	Const	CtrlT	As	Shortcut
[JScript]	public	var	CtrlT	:	Shortcut;

Description

The shortcut keys CTRL+T.

mmmmm) ToString

[C#]	public	const	Shortcut	CtrlU;	
[C++]	public:	const	Shortcut	CtrlU;	
[VB]	Public	Const	CtrlU	As	Shortcut
[JScript]	public	var	CtrlU	:	Shortcut;

Description

The shortcut keys CTRL+U The shorcut keys CTRL+U

nnnnn) ToString

[C#]	public	const	Shortcut	CtrlV;
[C++]	public:	const	Shortcut	CtrlV;

[VB]	Public	Const	CtrlV	As	Shortcut
[JScript]	public	var	CtrlV	:	Shortcut;

Description

The shortcut keys CTRL+V.

ooooo) ToString

[C#]	public	const	Shortcut	CtrlW;	
[C++]	public:	const	Shortcut	CtrlW;	
[VB]	Public	Const	CtrlW	As	Shortcut
[JScript]	public	var	CtrlW	:	Shortcut;

Description

The shortcut keys CTRL+W.

ppppp) ToString

[C#]	public	const	Shortcut	CtrlX;	
[C++]	public:	const	Shortcut	CtrlX;	
[VB]	Public	Const	CtrlX	As	Shortcut
[JScript]	public	var	CtrlX	:	Shortcut;

Description

The shortcut keys CTRL+X.

qqqqq) ToString

[C#]	public	const	Shortcut	CtrlY;
[C++]	public:	const	Shortcut	CtrlY;
[VB]	Public	Const	CtrlY As	Shortcut
[JScript]	public	var	CtrlY :	Shortcut;

Description

The shortcut keys CTRL+Y.

rrrrr) ToString

[C#]	public	const	Shortcut	CtrlZ;
[C++]	public:	const	Shortcut	CtrlZ;
[VB]	Public	Const	CtrlZ As	Shortcut
[JScript]	public	var	CtrlZ :	Shortcut;

Description

The shortcut keys CTRL+Z.

sssss) ToString

[C#]	public	const	Shortcut	Del;
[C++]	public:	const	Shortcut	Del;
[VB]	Public	Const	Del As	Shortcut
[JScript]	public	var	Del :	Shortcut;

Description

The shortcut key DELETE.

ttttt) ToString

[C#]	public	const	Shortcut	F1;
------	--------	-------	----------	-----

[C++]	public:	const	Shortcut	F1;
-------	---------	-------	----------	-----

[VB]	Public	Const	F1	As	Shortcut
------	--------	-------	----	----	----------

[JScript]	public	var	F1	:	Shortcut;
-----------	--------	-----	----	---	-----------

Description

The shortcut key F1.

uuuuu)ToString

[C#]	public	const	Shortcut	F10;
------	--------	-------	----------	------

[C++]	public:	const	Shortcut	F10;
-------	---------	-------	----------	------

[VB]	Public	Const	F10	As	Shortcut
------	--------	-------	-----	----	----------

[JScript]	public	var	F10	:	Shortcut;
-----------	--------	-----	-----	---	-----------

Description

The shortcut key F10.

vvvvv) ToString

[C#]	public	const	Shortcut	F11;
------	--------	-------	----------	------

[C++]	public:	const	Shortcut	F11;
-------	---------	-------	----------	------

1 [VB] Public Const F11 As Shortcut

2 [JScript] public var F11 : Shortcut;

3
4 *Description*

5 The shortcut key F11.

6 *wwwwww)ToString*

7
8 [C#] public const Shortcut F12;

9 [C++] public: const Shortcut F12;

10 [VB] Public Const F12 As Shortcut

11 [JScript] public var F12 : Shortcut;

12
13 *Description*

14 The shortcut key F12.

15 *xxxxx) ToString*

16
17 [C#] public const Shortcut F2;

18 [C++] public: const Shortcut F2;

19 [VB] Public Const F2 As Shortcut

20 [JScript] public var F2 : Shortcut;

21
22 *Description*

23 The shortcut key F2.

yyyyy) ToString

[C#]	public	const	Shortcut	F3;	
[C++]	public:	const	Shortcut	F3;	
[VB]	Public	Const	F3	As	Shortcut
[JScript]	public	var	F3	:	Shortcut;

Description

The shortcut key F3.

zzzzz) ToString

[C#]	public	const	Shortcut	F4;	
[C++]	public:	const	Shortcut	F4;	
[VB]	Public	Const	F4	As	Shortcut
[JScript]	public	var	F4	:	Shortcut;

Description

The shortcut key F4.

aaaaaa) ToString

[C#]	public	const	Shortcut	F5;	
[C++]	public:	const	Shortcut	F5;	
[VB]	Public	Const	F5	As	Shortcut
[JScript]	public	var	F5	:	Shortcut;

Description

The shortcut key F5.

bbbbbb)ToString

[C#]	public	const	Shortcut	F6;
------	--------	-------	----------	-----

[C++]	public:	const	Shortcut	F6;
-------	---------	-------	----------	-----

[VB]	Public	Const	F6	As	Shortcut
------	--------	-------	----	----	----------

[JScript]	public	var	F6	:	Shortcut;
-----------	--------	-----	----	---	-----------

Description

The shortcut key F6.

cccccc)ToString

[C#]	public	const	Shortcut	F7;
------	--------	-------	----------	-----

[C++]	public:	const	Shortcut	F7;
-------	---------	-------	----------	-----

[VB]	Public	Const	F7	As	Shortcut
------	--------	-------	----	----	----------

[JScript]	public	var	F7	:	Shortcut;
-----------	--------	-----	----	---	-----------

Description

The shortcut key F7.

dddddd)ToString

[C#]	public	const	Shortcut	F8;
------	--------	-------	----------	-----

[C++]	public:	const	Shortcut	F8;
-------	---------	-------	----------	-----

[VB]	Public	Const	F8	As	Shortcut
[JScript]	public	var	F8	:	Shortcut;

Description

The shortcut key F8.

eeeeee)ToString

[C#]	public	const		Shortcut	F9;
[C++]	public:	const		Shortcut	F9;
[VB]	Public	Const	F9	As	Shortcut
[JScript]	public	var	F9	:	Shortcut;

Description

The shortcut key F9.

ffffff) ToString

[C#]	public	const		Shortcut	Ins;
[C++]	public:	const		Shortcut	Ins;
[VB]	Public	Const	Ins	As	Shortcut
[JScript]	public	var	Ins	:	Shortcut;

Description

The shortcut key INSERT.

ggggg)ToString

[C#]	public	const	Shortcut	None;
[C++]	public:	const	Shortcut	None;
[VB]	Public	Const	None	As Shortcut
[JScript]	public	var	None	: Shortcut;

Description

No shortcut key is associated with the menu item.

hhhhh)ToString

[C#]	public	const	Shortcut	ShiftDel;
[C++]	public:	const	Shortcut	ShiftDel;
[VB]	Public	Const	ShiftDel	As Shortcut
[JScript]	public	var	ShiftDel	: Shortcut;

Description

The shortcut keys SHIFT+DELETE.

iiii) ToString

[C#]	public	const	Shortcut	ShiftF1;
[C++]	public:	const	Shortcut	ShiftF1;
[VB]	Public	Const	ShiftF1	As Shortcut
[JScript]	public	var	ShiftF1	: Shortcut;

Description

The shortcut keys SHIFT+F1.

jjjjj) ToString

[C#]	public	const	Shortcut	ShiftF10;
[C++]	public:	const	Shortcut	ShiftF10;
[VB]	Public	Const	ShiftF10	As Shortcut
[JScript]	public	var	ShiftF10	: Shortcut;

Description

The shortcut keys SHIFT+F10.

kkkkkk)ToString

[C#]	public	const	Shortcut	ShiftF11;
[C++]	public:	const	Shortcut	ShiftF11;
[VB]	Public	Const	ShiftF11	As Shortcut
[JScript]	public	var	ShiftF11	: Shortcut;

Description

The shortcut keys SHIFT+F11.

lllll) ToString

[C#]	public	const	Shortcut	ShiftF12;
[C++]	public:	const	Shortcut	ShiftF12;

[VB]	Public	Const	ShiftF12	As	Shortcut
[JScript]	public	var	ShiftF12	:	Shortcut;

Description

The shortcut keys SHIFT+F12.

mmmmmm)ToString

[C#]	public	const	Shortcut	ShiftF2;	
[C++]	public:	const	Shortcut	ShiftF2;	
[VB]	Public	Const	ShiftF2	As	Shortcut
[JScript]	public	var	ShiftF2	:	Shortcut;

Description

The shortcut keys SHIFT+F2.

nnnnnn)ToString

[C#]	public	const	Shortcut	ShiftF3;	
[C++]	public:	const	Shortcut	ShiftF3;	
[VB]	Public	Const	ShiftF3	As	Shortcut
[JScript]	public	var	ShiftF3	:	Shortcut;

Description

The shortcut keys SHIFT+F3.

ooooooo)ToString

[C#]	public	const	Shortcut	ShiftF4;
[C++]	public:	const	Shortcut	ShiftF4;
[VB]	Public	Const	ShiftF4	As Shortcut
[JScript]	public	var	ShiftF4	: Shortcut;

Description

The shortcut keys SHIFT+F4.

ppppppp)ToString

[C#]	public	const	Shortcut	ShiftF5;
[C++]	public:	const	Shortcut	ShiftF5;
[VB]	Public	Const	ShiftF5	As Shortcut
[JScript]	public	var	ShiftF5	: Shortcut;

Description

The shortcut keys SHIFT+F5.

qqqqqqq)ToString

[C#]	public	const	Shortcut	ShiftF6;
[C++]	public:	const	Shortcut	ShiftF6;
[VB]	Public	Const	ShiftF6	As Shortcut
[JScript]	public	var	ShiftF6	: Shortcut;

Description

The shortcut keys SHIFT+F6.

rrrrrr) ToString

[C#]	public	const	Shortcut	ShiftF7;
[C++]	public:	const	Shortcut	ShiftF7;
[VB]	Public	Const	ShiftF7	As Shortcut
[JScript]	public	var	ShiftF7	: Shortcut;

Description

The shortcut keys SHIFT+F7.

ssssss) ToString

[C#]	public	const	Shortcut	ShiftF8;
[C++]	public:	const	Shortcut	ShiftF8;
[VB]	Public	Const	ShiftF8	As Shortcut
[JScript]	public	var	ShiftF8	: Shortcut;

Description

The shortcut keys SHIFT+F8.

tttttt) ToString

[C#]	public	const	Shortcut	ShiftF9;
[C++]	public:	const	Shortcut	ShiftF9;

[VB]	Public	Const	ShiftF9	As	Shortcut
[JScript]	public	var	ShiftF9	:	Shortcut;

Description

The shortcut keys SHIFT+F9.

uuuuuu)ToString

[C#]	public	const	Shortcut	ShiftIns;	
[C++]	public:	const	Shortcut	ShiftIns;	
[VB]	Public	Const	ShiftIns	As	Shortcut
[JScript]	public	var	ShiftIns	:	Shortcut;

Description

The shortcut keys SHIFT+INSERT.

SizeGripStyle enumeration (System.Windows.Forms)

a) ToString

Description

Specifies the style of the sizing grip on a **System.Windows.Forms.Form** .

Use the members of this enumeration to set the value of the **System.Windows.Forms.Form.SizeGripStyle** property of the **System.Windows.Forms.Form** .

b) ToString

[C#]	public	const	SizeGripStyle	Auto;
[C++]	public:	const	SizeGripStyle	Auto;
[VB]	Public	Const	Auto As	SizeGripStyle
[JScript]	public	var	Auto :	SizeGripStyle;

Description

The sizing grip is automatically displayed when needed.

c) ToString

[C#]	public	const	SizeGripStyle	Hide;
[C++]	public:	const	SizeGripStyle	Hide;
[VB]	Public	Const	Hide As	SizeGripStyle
[JScript]	public	var	Hide :	SizeGripStyle;

Description

The sizing grip is hidden.

d) ToString

[C#]	public	const	SizeGripStyle	Show;
[C++]	public:	const	SizeGripStyle	Show;
[VB]	Public	Const	Show As	SizeGripStyle
[JScript]	public	var	Show :	SizeGripStyle;

Description

The sizing grip is always shown on the form.

SortOrder enumeration (System.Windows.Forms)

a) ToString

Description

Specifies how items in a list are sorted.

Use the members of this enumeration to set the value of the **System.Windows.Forms.ListView.Sorting** property of the **System.Windows.Forms.ListView** control.

b) ToString

[C#]	public	const	SortOrder	Ascending;
[C++]	public:	const	SortOrder	Ascending;
[VB]	Public	Const	Ascending As	SortOrder
[JScript]	public	var	Ascending :	SortOrder;

Description

The items are sorted in ascending order.

c) ToString

[C#]	public	const	SortOrder	Descending;
[C++]	public:	const	SortOrder	Descending;

[VB]	Public	Const	Descending	As	SortOrder
[JScript]	public	var	Descending	:	SortOrder;

Description

The items are sorted in descending order.

d) ToString

[C#]	public	const		SortOrder	None;
[C++]	public:	const		SortOrder	None;
[VB]	Public	Const	None	As	SortOrder
[JScript]	public	var	None	:	SortOrder;

Description

The items are not sorted.

Splitter class (System.Windows.Forms)

a) ToString

Description

Represents a splitter control that provides the ability for the user to resize docked controls.

The **System.Windows.Forms.Splitter** control enables you to resize controls that are docked to the edges of the **System.Windows.Forms.Splitter** control at runtime. When the user passes the mouse pointer over the **System.Windows.Forms.Splitter** control, the cursor changes to indicate that the controls docked to the **System.Windows.Forms.Splitter** control can be resized. The **System.Windows.Forms.Splitter** control enables the user to resize the docked control that is immediately before it in the docking order.

Therefore, to enable the user to resize a docked control, dock the control you want the user to be able to resize to an edge of a container, and then dock a splitter to the same side of that container. For example, to create a window similar to Windows Explorer, add a **System.Windows.Forms.TreeView** control to a form and set its **System.Windows.Forms.Control.Dock** property to **DockStyle.Left**. Add a **System.Windows.Forms.Splitter** control to the form and set its **System.Windows.Forms.Control.Dock** property to **DockStyle.Left** as well. To complete the form layout, add a **System.Windows.Forms.ListView** control and set its **System.Windows.Forms.Control.Dock** property to **DockStyle.Fill** to have the **System.Windows.Forms.ListView** occupy the remaining space on the form. At runtime, the user can then resize the width of the **System.Windows.Forms.TreeView** control (as well as the **System.Windows.Forms.ListView** control) by moving the **System.Windows.Forms.Splitter** control.

b) Splitter

Example Syntax:

c) ToString

[C#]	public	Splitter();
[C++]	public:	Splitter();
[VB]	Public	Sub New()
[JScript]	public	function Splitter();

Description

Initializes a new instance of the **System.Windows.Forms.Splitter** class.

- d) *AccessibilityObject*
- e) *AccessibleDefaultActionDescription*
- f) *AccessibleDescription*
- g) *AccessibleName*
- h) *AccessibleRole*
- i) *AllowDrop*
- j) *ToString*

Description

- k) *Anchor*
- l) *ToString*

```
[C#]    public    override    AnchorStyles    Anchor    {get;    set;}
[C++]  public: __property virtual AnchorStyles get_Anchor();public: __property
virtual                                void                                set_Anchor(AnchorStyles);
[VB]    Overrides    Public    Property    Anchor    As    AnchorStyles
[JScript] public function get Anchor() : AnchorStyles;public function set
Anchor(AnchorStyles);
```

Description

The current value of the anchor property. The anchor property determines which edges of the control are anchored to the container's edges.

- m) *BackColor*
- n) *BackgroundImage*
- o) *ToString*

Description

- p) *BindingContext*
- q) *BorderStyle*
- r) *ToString*

Description

Gets or sets the style of border for the **System.Windows.Forms.Splitter** control.

- s) *Bottom*
- t) *Bounds*
- u) *CanFocus*
- v) *CanSelect*
- w) *Capture*
- x) *CausesValidation*
- y) *ClientRectangle*
- z) *ClientSize*
- aa) *CompanyName*
- bb) *Container*
- cc) *ContainsFocus*
- dd) *ContextMenu*
- ee) *Controls*
- ff) *Created*
- gg) *CreateParams*
- hh) *ToString*

Description

Returns the parameters needed to create the handle. Inheriting classes can override this to provide extra functionality. They should not, however, forget to call `base.CreateParams()` first to get the struct filled up with the basic info.

1 *ii) Cursor*

2 *jj) DataBindings*

3 *kk) DefaultImeMode*

4 *ll) ToString*

5
6
7 *Description*

8 Gets the default Input Method Editor (IME) mode supported by this control.

9 As implemented in the **System.Windows.Forms.Splitter** class, this property
10 always returns the **System.Windows.Forms.ImeMode.Disable** value.

11 *mm) DefaultSize*

12 *nn) ToString*

13 [C#] protected override Size DefaultSize {get;}

14 [C++] protected: __property virtual Size get_DefaultSize();

15 [VB] Overrides Protected ReadOnly Property DefaultSize As Size

16 [JScript] protected function get DefaultSize() : Size;

17
18 *Description*

19 Gets the default size of the control.

1 *oo) DesignMode*
2 *pp) DisplayRectangle*
3 *qq) Disposing*
4 *rr) Dock*
5 *ss) ToString*

6
7
8 *Description*

9 *tt) Enabled*
10 *uu) Events*
11 *vv) Focused*
12 *ww) Font*
13 *xx) ToString*

14
15
16 *Description*

17
18 *yy) FontHeight*
19 *zz) ForeColor*
20 *aaa) ToString*

21
22
23
24 *Description*
25

1 *bbb) Handle*

2 *ccc) HasChildren*

3 *ddd) Height*

4 *eee) ImeMode*

5 *fff) ToString*

6
7
8 *Description*

9 Gets or sets the Input Method Editor (IME) mode supported by this control.

10 *ggg) InvokeRequired*

11 *hhh) IsAccessible*

12 *iii) IsDisposed*

13 *jjj) IsHandleCreated*

14 *kkk) Left*

15 *lll) Location*

16 *mmm) MinExtra*

17 *nnn) ToString*

18
19
20
21 *Description*

22 Gets or sets the minimum distance that must remain between the
23 **System.Windows.Forms.Splitter** control and the edge of the opposite side of
24 the container (or the closest control docked to that side).

25 For a **System.Windows.Forms.Splitter** control docked horizontally (a
 System.Windows.Forms.Splitter control docked to the top or bottom of a
 container), the minimum height of the area of the container reserved for

undocked controls is this value minus the height of the **System.Windows.Forms.Splitter** control. For a **System.Windows.Forms.Splitter** control docked vertically (a **System.Windows.Forms.Splitter** control docked to the left or right of a container), the minimum width of the area of the container reserved for undocked controls is this value minus the width of the **System.Windows.Forms.Splitter** control. The user cannot move the splitter past the limit specified by this property.

ooo) MinSize

ppp) ToString

[C#] public int MinSize {get; set;}

[C++] public: __property int get_MinSize();public: __property void
set_MinSize(int);

[VB] Public Property MinSize As Integer

[JScript] public function get MinSize() : int;public function set MinSize(int);

Description

Gets or sets the minimum distance that must remain between the **System.Windows.Forms.Splitter** control and the container edge that the control is docked to.

For a **System.Windows.Forms.Splitter** control docked horizontally (a **System.Windows.Forms.Splitter** control docked to the top or bottom of a container), this value is the minimum height of the resizable control. For a **System.Windows.Forms.Splitter** control docked vertically (a **System.Windows.Forms.Splitter** control docked to the left or right of a container), this value is the minimum width of the resizable control. The user cannot move the splitter past the limit specified by this property.

qqq) Name
rrr) Parent
sss) ProductName
ttt) ProductVersion
uuu) RecreatingHandle
vvv) Region
www) RenderRightToLeft
xxx) ResizeRedraw
yyy) Right
zzz) RightToLeft
aaaa) ShowFocusCues
bbbb) ShowKeyboardCues
cccc) Site
dddd) Size
eeee) SplitPosition
ffff) ToString

Description

Gets or sets the distance between the **System.Windows.Forms.Splitter** control and the container edge that the control is docked to.

For a horizontal **System.Windows.Forms.Splitter** control (a **System.Windows.Forms.Splitter** control docked to the top or bottom of a container), this value is the height of the resizable control. For a vertical **System.Windows.Forms.Splitter** control (a **System.Windows.Forms.Splitter** control docked to the left or right of a

container), this value is the width of the resizable control. You can use the **System.Windows.Forms.Splitter.SplitPosition** property in an event handler for the **System.Windows.Forms.Splitter.SplitterMoved** or **System.Windows.Forms.Splitter.SplitterMoving** events to determine the size of the control that the **System.Windows.Forms.Splitter** control is docked to and limit its width to a specified maximum width (or height if a horizontally aligned **System.Windows.Forms.Splitter** control).

gggg) TabIndex

hhhh) TabStop

iiii) ToString

Description

jjjj) Tag

kkkk) Text

llll) ToString

Description

mmmm)Top

nnnn) TopLevelControl

oooo) Visible

pppp) Width

qqqq) WindowTarget

rrrr) ToString

Description

ssss) ToString

Description

tttt) ToString

[C#] public new event KeyPressEventHandler KeyPress;

[C++] public: __event KeyPressEventHandler* KeyPress;

[VB] Shadows Public Event KeyPress As KeyPressEventHandler

Description

uuuu) ToString

```
[C#]      public      new      event      KeyEventHandler      KeyUp;
[C++]      public:      __event      KeyEventHandler*      KeyUp;
[VB]      Shadows      Public      Event      KeyUp      As      KeyEventHandler
```

Description

vvvv) ToString

Description

www)ToString

Description

Occurs when the **System.Windows.Forms.Splitter** control is moved.

You can create an event handler for the **System.Windows.Forms.Splitter.SplitterMoved** to perform resize validation in your application. For example, if a **System.Windows.Forms.Splitter** control is docked to the edges of a **System.Windows.Forms.TreeView** control and a **System.Windows.Forms.ListView** control, you can write code in the **System.Windows.Forms.Splitter.SplitterMoved** event to determine if the minimum and/or maximum size for both the **System.Windows.Forms.TreeView** and the **System.Windows.Forms.ListView** controls has been exceeded and resize the controls to their minimum or maximum size.

xxxx) *ToString*

```
[C#]      public      event      SplitterEventHandler      SplitterMoving;
[C++]     public:     __event     SplitterEventHandler*      SplitterMoving;
[VB]      Public      Event      SplitterMoving      As      SplitterEventHandler
```

Description

Occurs when the **System.Windows.Forms.Splitter** control is in the process of moving.

You can create an event handler for the **System.Windows.Forms.Splitter.SplitterMoving** to perform resize validation in your application. For example, if a **System.Windows.Forms.Splitter** control is docked to the edges of a **System.Windows.Forms.TreeView** control and a **System.Windows.Forms.ListView** control, you can write code in the **System.Windows.Forms.Splitter.SplitterMoved** event to determine if the minimum and/or maximum size for both the **System.Windows.Forms.TreeView** and the **System.Windows.Forms.ListView** controls has been exceeded and restrict the resizing of the controls to their minimum or maximum size.

yyyy) *OnKeyDown*

```
[C#]      protected      override      void      OnKeyDown(KeyEventArgs      e);
[C++]     protected:      void      OnKeyDown(KeyEventArgs*      e);
[VB]      Overrides      Protected      Sub      OnKeyDown(ByVal e As KeyEventArgs)
[JScript] protected      override      function      OnKeyDown(e : KeyEventArgs);
```

Description

zzzz) *OnMouseDown*

[C#] protected override void OnMouseDown(MouseEventArgs e);
[C++] protected: void OnMouseDown(MouseEventArgs* e);
[VB] Overrides Protected Sub OnMouseDown(ByVal e As MouseEventArgs)
[JScript] protected override function OnMouseDown(e : MouseEventArgs);

Description

aaaaa) *OnMouseMove*

[C#] protected override void OnMouseMove(MouseEventArgs e);
[C++] protected: void OnMouseMove(MouseEventArgs* e);
[VB] Overrides Protected Sub OnMouseMove(ByVal e As MouseEventArgs)
[JScript] protected override function OnMouseMove(e : MouseEventArgs);

Description

bbbbbb) *OnMouseUp*

[C#] protected override void OnMouseUp(MouseEventArgs e);
[C++] protected: void OnMouseUp(MouseEventArgs* e);
[VB] Overrides Protected Sub OnMouseUp(ByVal e As MouseEventArgs)
[JScript] protected override function OnMouseUp(e : MouseEventArgs);

Description

cccc) OnSplitterMoved

[C#] protected virtual void OnSplitterMoved(SplitterEventArgs sevent);
[C++] protected: virtual void OnSplitterMoved(SplitterEventArgs* sevent);
[VB] Overridable Protected Sub OnSplitterMoved(ByVal sevent As SplitterEventArgs)
[JScript] protected function OnSplitterMoved(sevent : SplitterEventArgs);

Description

Raises the **System.Windows.Forms.Splitter.SplitterMoved** event.

Raising an event invokes the event handler through a delegate. For more information, see . A **System.Windows.Forms.SplitterEventArgs** that contains the event data.

dddd) OnSplitterMoving

[C#] protected virtual void OnSplitterMoving(SplitterEventArgs sevent);
[C++] protected: virtual void OnSplitterMoving(SplitterEventArgs* sevent);
[VB] Overridable Protected Sub OnSplitterMoving(ByVal sevent As SplitterEventArgs)
[JScript] protected function OnSplitterMoving(sevent : SplitterEventArgs);

Description

Raises the **System.Windows.Forms.Splitter.SplitterMoving** event.

Raising an event invokes the event handler through a delegate. For more information, see . A **System.Windows.Forms.SplitterEventArgs** that contains the event data.

eeee) PreFilterMessage

[C#] public bool PreFilterMessage(ref Message m);
[C++] public: __sealed bool PreFilterMessage(Message* m);
[VB] NotOverridable Public Function PreFilterMessage(ByRef m As Message) As Boolean
[JScript] public function PreFilterMessage(m : Message) : Boolean;

Description

ffff) SetBoundsCore

[C#] protected override void SetBoundsCore(int x, int y, int width, int height, BoundsSpecified specified);
[C++] protected: void SetBoundsCore(int x, int y, int width, int height, BoundsSpecified specified);
[VB] Overrides Protected Sub SetBoundsCore(ByVal x As Integer, ByVal y As Integer, ByVal width As Integer, ByVal height As Integer, ByVal specified As BoundsSpecified)
[JScript] protected override function SetBoundsCore(x : int, y : int, width : int, height : int, specified : BoundsSpecified);

Description

ggggg) ToString

[C#]	public	override	string	ToString();
[C++]	public:	String*	ToString();	
[VB]	Overrides	Public	Function	ToString() As String
[JScript]	public	override	function	ToString() : String;

Description

Returns a string representation for this control.

Return Value: String Returns a string representation for this control.

SplitterEventArgs class (System.Windows.Forms)

a) WndProc

Description

Provides data for **System.Windows.Forms.Splitter.SplitterMoving** and the **System.Windows.Forms.Splitter.SplitterMoved** events.

The **System.Windows.Forms.Splitter.SplitterMoving** event occurs when the user is moving the **System.Windows.Forms.Splitter** control. The and **System.Windows.Forms.Splitter.SplitterMoved** events occur when the user finishes moving the **System.Windows.Forms.Splitter** control. The **System.Windows.Forms.SplitterEventArgs** class specifies the position of the mouse pointer and the position of the upper-left corner of the **System.Windows.Forms.Splitter** control.

b) SplitterEventArgs

Example Syntax:

c) *WndProc*

```
[C#] public SplitterEventArgs(int x, int y, int splitX, int splitY);  
[C++] public: SplitterEventArgs(int x, int y, int splitX, int splitY);  
[VB] Public Sub New(ByVal x As Integer, ByVal y As Integer, ByVal splitX As  
Integer, ByVal splitY As Integer)  
[JScript] public function SplitterEventArgs(x : int, y : int, splitX : int, splitY : int);
```

Description

Initializes an instance of the **System.Windows.Forms.SplitterEventArgs** class with the specified coordinates of the mouse pointer and the coordinates of the upper-left corner of the **System.Windows.Forms.Splitter** control. The x-coordinate of the mouse pointer (in client coordinates). The y-coordinate of the mouse pointer (in client coordinates). The x-coordinate of the upper-left corner of the **System.Windows.Forms.Splitter** (in client coordinates). The y-coordinate of the upper-left corner of the **System.Windows.Forms.Splitter** (in client coordinates).

d) *SplitX*

e) *WndProc*

```
[C#] public int SplitX {get; set;}  
[C++] public: __property int get_SplitX();public: __property void set_SplitX(int);  
[VB] Public Property SplitX As Integer  
[JScript] public function get SplitX() : int;public function set SplitX(int);
```

Description

Gets or sets the x-coordinate of the upper-left corner of the **System.Windows.Forms.Splitter** (in client coordinates).

You can use this property along with the

System.Windows.Forms.SplitterEventArgs.SplitY property of this class to determine the position of the **System.Windows.Forms.Splitter** control when the **System.Windows.Forms.Splitter.SplitterMoving** and **System.Windows.Forms.Splitter.SplitterMoved** events are raised.

f) *SplitY*

g) *WndProc*

[C#] public int SplitY {get; set;}

[C++] public: __property int get_SplitY();public: __property void set_SplitY(int);

[VB] Public Property SplitY As Integer

[JScript] public function get SplitY() : int;public function set SplitY(int);

Description

Gets or sets the y-coordinate of the upper-left corner of the **System.Windows.Forms.Splitter** (in client coordinates).

You can use this property along with the

System.Windows.Forms.SplitterEventArgs.SplitX property of this class to determine the position of the **System.Windows.Forms.Splitter** control when the **System.Windows.Forms.Splitter.SplitterMoving** and **System.Windows.Forms.Splitter.SplitterMoved** events are raised.

h) *X*

i) *WndProc*

[C#] public int X {get;}

[C++] public: __property int get_X();

[VB] Public ReadOnly Property X As Integer

[JScript] public function get X() : int;

Description

Gets the x-coordinate of the mouse pointer (in client coordinates).

You can use this property along with the **System.Windows.Forms.SplitterEventArgs.Y** property of this class to determine the current location of the mouse pointer when the **System.Windows.Forms.Splitter.SplitterMoving** and **System.Windows.Forms.Splitter.SplitterMoved** events are raised.

j) *Y*

k) *WndProc*

[C#]	public	int	Y	{get;}
[C++]	public:	__property	int	get_Y();
[VB]	Public	ReadOnly	Property	Y As Integer
[JScript]	public	function	get	Y() : int;

Description

Gets the y-coordinate of the mouse pointer (in client coordinates).

You can use this property along with the **System.Windows.Forms.SplitterEventArgs.X** property of this class to determine the current location of the mouse pointer when the **System.Windows.Forms.Splitter.SplitterMoving** and **System.Windows.Forms.Splitter.SplitterMoved** events are raised.

SplitterEventHandler delegate (System.Windows.Forms)

a) *ToString*

Description

Represents the method that will handle the **System.Windows.Forms.Splitter.SplitterMoving** and **System.Windows.Forms.Splitter.SplitterMoved** events of a **System.Windows.Forms.Splitter**. The source of the event. A **System.Windows.Forms.SplitterEventArgs** that contains the event data.

When you create a(n) **System.Windows.Forms.SplitterEventHandler** delegate, you identify the method that will handle the event. To associate the event with your event handler, add an instance of the delegate to the event. The event handler is called whenever the event occurs, unless you remove the delegate. For more information about event handler delegates, see .

AxHost.State class (System.Windows.Forms)

a) *ToString*

Description

Encapsulates the persisted state of an ActiveX control.

The **System.Windows.Forms.AxHost.State** object can be created by retrieving the **System.Windows.Forms.AxHost.OcxState** property of an **System.Windows.Forms.AxHost** object, or by reading the control's state from a data stream.

b) *AxHost.State*

Example Syntax:

c) *ToString*

```
[C#] public AxHost.State(Stream ms, int storageType, bool manualUpdate, string  
licKey);
```

```
[C++] public: State(Stream* ms, int storageType, bool manualUpdate, String*  
licKey);
```

```
[VB] Public Sub New(ByVal ms As Stream, ByVal storageType As Integer,  
ByVal manualUpdate As Boolean, ByVal licKey As String)
```

1 [JScript] public function AxHost.State(ms : Stream, storageType : int,
2 manualUpdate : Boolean, licKey : String);

3 *d) ISerializable.GetObjectData*

5 [C#] void ISerializable.GetObjectData(SerializationInfo si, StreamingContext
6 context);

7 [C++] void ISerializable::GetObjectData(SerializationInfo* si, StreamingContext
8 context);

9 [VB] Sub GetObjectData(ByVal si As SerializationInfo, ByVal context As
10 StreamingContext) Implements ISerializable.GetObjectData

11 [JScript] function ISerializable.GetObjectData(si : SerializationInfo, context :
12 StreamingContext);

13 StatusBar class (System.Windows.Forms)

14 *a) ToString*

17 *Description*

18 Represents a Windows status bar control.

19 Typically a **System.Windows.Forms.StatusBar** control consists of
20 **System.Windows.Forms.StatusBarPanel** objects, each of which typically
21 displays text and/or an icon. You can also provide owner-drawn panels to
22 provide custom panels such as a progress bar or a series of images that displays
23 the state of your application. A **System.Windows.Forms.StatusBar** control
24 typically displays information about an object being viewed on a
25 **System.Windows.Forms.Form**, the object's components, or contextual
information that relates to that object's operation within your application.

24 *b) StatusBar*

25 *Example Syntax:*

c) *ToString*

[C#]	public	StatusBar();
[C++]	public:	StatusBar();
[VB]	Public Sub	New()
[JScript]	public function	StatusBar();

Description

Initializes a new instance of the **System.Windows.Forms.StatusBar** class.

The default **System.Windows.Forms.StatusBar** has no panels.

d) *AccessibilityObject*

e) *AccessibleDefaultActionDescription*

f) *AccessibleDescription*

g) *AccessibleName*

h) *AccessibleRole*

i) *AllowDrop*

j) *Anchor*

k) *BackColor*

l) *ToString*

Description

The background color of this control. This is an ambient property and will always return a non-null value.

1 m) *BackgroundImage*

2 n) *ToString*

3
4 [C#] public override Image BackgroundImage {get; set;}

5 [C++] public: __property virtual Image* get_BackgroundImage();public:

6 __property virtual void set_BackgroundImage(Image*);

7 [VB] Overrides Public Property BackgroundImage As Image

8 [JScript] public function get BackgroundImage() : Image;public function set

9 BackgroundImage(Image);

10
11 *Description*

12 Gets or sets the image rendered on the background of the
13 **System.Windows.Forms.StatusBar** control.

- o) *BindingContext*
- p) *Bottom*
- q) *Bounds*
- r) *CanFocus*
- s) *CanSelect*
- t) *Capture*
- u) *CausesValidation*
- v) *ClientRectangle*
- w) *ClientSize*
- x) *CompanyName*
- y) *Container*
- z) *ContainsFocus*
- aa) *ContextMenu*
- bb) *Controls*
- cc) *Created*
- dd) *CreateParams*
- ee) *ToString*

Description

Gets the **System.Windows.Forms.CreateParams** object used to create the handle for this control.

ff) Cursor

gg) DataBindings

hh) DefaultImeMode

ii) ToString

Description

Gets the default Input Method Editor (IME) mode supported by this control.

As implemented in the **System.Windows.Forms.StatusBar** class, this property always returns the **System.Windows.Forms.ImeMode.Disable** value.

jj) DefaultSize

kk) ToString

[C#] protected override Size DefaultSize {get;}

[C++] protected: __property virtual Size get_DefaultSize();

[VB] Overrides Protected ReadOnly Property DefaultSize As Size

[JScript] protected function get DefaultSize() : Size;

Description

Gets the default size of the control.

1 *ll) DesignMode*

2 *mm) DisplayRectangle*

3 *nn) Disposing*

4 *oo) Dock*

5 *pp) ToString*

6
7
8 *Description*

9 Gets or sets the docking behavior of the **System.Windows.Forms.StatusBar**
10 control.

11 *qq) Enabled*

12 *rr) Events*

13 *ss) Focused*

14 *tt) Font*

15 *uu) ToString*

16
17
18 *Description*

19 Gets or sets the font the **System.Windows.Forms.StatusBar** control will use
20 to display information.

1 *vv) FontHeight*

2 *ww) ForeColor*

3 *xx) ToString*

4
5
6 *Description*

7 Gets or sets the forecolor for the control.

8 *yy) Handle*

9 *zz) HasChildren*

10 *aaa) Height*

11 *bbb) ImeMode*

12 *ccc) ToString*

13
14
15 *Description*

16 Gets or sets the Input Method Editor (IME) mode supported by this control.

ddd) *InvokeRequired*

eee) *IsAccessible*

fff) *IsDisposed*

ggg) *IsHandleCreated*

hhh) *Left*

iii) *Location*

jjj) *Name*

kkk) *Panels*

lll) *ToString*

Description

Gets the collection of **System.Windows.Forms.StatusBar** panels contained within the control.

The **System.Windows.Forms.StatusBar** control can display a number of panels provide information to the user of your application. For example, a panel could display the current time or the progress of a file download. Each panel displayed by the **System.Windows.Forms.StatusBar** control is an instance of the **System.Windows.Forms.StatusBarPanel** class. The **System.Windows.Forms.StatusBar.Panels** property enables you to obtain a reference to the collection of **System.Windows.Forms.StatusBarPanel** objects that are currently stored in the **System.Windows.Forms.StatusBar** control. With this reference, you can add panels, remove panels, access a specific panel within the collection, and obtain a count of the panels in the **System.Windows.Forms.StatusBar** control. For more information on the tasks that can be performed with the panel collection, see the **System.Windows.Forms.StatusBar.StatusBarPanelCollection** class reference topics.

mmm) *Parent*

nnn) *ProductName*

ooo) *ProductVersion*

ppp) *RecreatingHandle*

qqq) *Region*

rrr) *RenderRightToLeft*

sss) *ResizeRedraw*

ttt) *Right*

uuu) *RightToLeft*

vvv) *ShowFocusCues*

www) *ShowKeyboardCues*

xxx) *ShowPanels*

yyy) *ToString*

Description

Gets or sets a value indicating whether any panels that have been added to the **System.Windows.Forms.StatusBar** control are displayed.

By default, the **System.Windows.Forms.StatusBar** control displays the value of its **System.Windows.Forms.Control.Text** property without any panels.

When this property is set to **true**, any

System.Windows.Forms.StatusBarPanel objects specified in the **System.Windows.Forms.StatusBar** control are displayed. No panels are initially created when you instantiate an instance of the

System.Windows.Forms.StatusBar class. You can add panels to a

System.Windows.Forms.StatusBar control by using the

System.Windows.Forms.StatusBar.StatusBarPanelCollection.Add(System.String) method of the

System.Windows.Forms.StatusBar.StatusBarPanelCollection class. This

collection class can be accessed through the
System.Windows.Forms.StatusBar.Panels property of
System.Windows.Forms.StatusBar .

zzz) Site

aaaa) Size

bbbb) SizingGrip

cccc) ToString

Description

Gets or sets a value indicating whether a sizing grip is displayed in the lower-right corner of the **System.Windows.Forms.StatusBar** control.

You can use this property to display a sizing grip to provide an indication to the user when a form is resizable. If the **System.Windows.Forms.Form.FormBorderStyle** property of your **System.Windows.Forms.Form** is set to a border style that is not resizable, such as **FormBorderStyle.Fixed3D** or **FormBorderStyle.Dialog** , you should set the **System.Windows.Forms.StatusBar.SizingGrip** property to **false** to prevent the user from thinking that the form can be resized.

dddd) TabIndex

eeee) TabStop

ffff) ToString

Description

Gets or sets a value indicating whether the user will be able to tab to the **System.Windows.Forms.StatusBar** .

1 *gggg) Tag*

2 *hhhh) Text*

3 *iiii) ToString*

4
5
6 *Description*

7 Gets or sets the status bar text.

8 *jiii) Top*

9 *kkkk) TopLevelControl*

10 *llll) Visible*

11 *mmmm)Width*

12 *nnnn) WindowTarget*

13 *oooo) ToString*

14
15
16 *Description*

17 Occurs when a visual aspect of an owner-drawn
18 **System.Windows.Forms.StatusBar** control changes.

19 You can use this event to perform drawing operations in an owner-drawn
20 **System.Windows.Forms.StatusBar** control. For example, if you display an
21 owner-drawn **System.Windows.Forms.StatusBarPanel** object that displays a
22 progress bar, you can use this event to perform the drawing of the progress bar
23 on the panel. The data provided to the event through the
24 **System.Windows.Forms.StatusBarDrawItemEventArgs** object passed as
25 a parameter to the event handler enables you to determine the panel that needs
 to be drawn and the **System.Drawing.Graphics** object to use to draw to the
 panel.

pppp) ToString

Description

Occurs when a **System.Windows.Forms.StatusBarPanel** object on a **System.Windows.Forms.StatusBar** control is clicked.

You can use this event to perform tasks when a panel within a **System.Windows.Forms.StatusBar** control is clicked. The data provided to the event through the **System.Windows.Forms.StatusBarPanelClickEventArgs** object passed as a parameter to the event handler enables you to determine the **System.Windows.Forms.StatusBarPanel** object that was clicked by the user in order to perform tasks on the selected panel.

qqqq) CreateHandle

[C#]	protected	override	void	CreateHandle();
[C++]	protected:		void	CreateHandle();
[VB]	Overrides	Protected	Sub	CreateHandle()
[JScript]	protected	override	function	CreateHandle();

Description

rrrr) Dispose

[C#]	protected	override	void	Dispose(bool disposing);
[C++]	protected:		void	Dispose(bool disposing);
[VB]	Overrides	Protected	Sub	Dispose(ByVal disposing As Boolean)
[JScript]	protected	override	function	Dispose(disposing : Boolean);

Description

Disposes of the resources (other than memory) used by the **System.Windows.Forms.StatusBar** .

This method is called by the public **Dispose()** method and the **System.Object.Finalize** method.

ssss) OnDrawItem

[C#] protected virtual void OnDrawItem(StatusBarDrawItemEventArgs sbdievent);

[C++] protected: virtual void OnDrawItem(StatusBarDrawItemEventArgs* sbdievent);

[VB] Overridable Protected Sub OnDrawItem(ByVal sbdievent As StatusBarDrawItemEventArgs)

[JScript] protected function OnDrawItem(sbdievent : StatusBarDrawItemEventArgs);

Description

Raises the **System.Windows.Forms.StatusBar.OnDrawItem(System.Windows.Forms.StatusBarDrawItemEventArgs)** event.

Raising an event invokes the event handler through a delegate. For more information, see . A

System.Windows.Forms.StatusBarDrawItemEventArgs that contains the event data.

tttt) OnHandleCreated

[C#] protected override void OnHandleCreated(EventArgs e);

```

1 [C++]      protected:      void      OnHandleCreated(EventArgs*      e);
2 [VB] Overrides Protected Sub OnHandleCreated(ByVal e As EventArgs)
3 [JScript] protected override function OnHandleCreated(e : EventArgs);

```

Description

Raises the **System.Windows.Forms.Control.HandleCreated** event.

Raising an event invokes the event handler through a delegate. For more information, see . An **System.EventArgs** that contains the event data.

uuuu) OnHandleDestroyed

```

10 [C#]      protected      override      void      OnHandleDestroyed(EventArgs      e);
11 [C++]      protected:      void      OnHandleDestroyed(EventArgs*      e);
12 [VB] Overrides Protected Sub OnHandleDestroyed(ByVal e As EventArgs)
13 [JScript] protected override function OnHandleDestroyed(e : EventArgs);

```

Description

Raises the **System.Windows.Forms.Control.HandleDestroyed** event.

Raising an event invokes the event handler through a delegate. For more information, see . . An **System.EventArgs** that contains the event data.

vvvv) OnLayout

```

21 [C#]      protected      override      void      OnLayout(LayoutEventArgs      levent);
22 [C++]      protected:      void      OnLayout(LayoutEventArgs*      levent);
23 [VB] Overrides Protected Sub OnLayout(ByVal levent As LayoutEventArgs)
24 [JScript] protected override function OnLayout(levent : LayoutEventArgs);

```


Description

Raises the Layout event.

Raising an event invokes the event handler through a delegate. For more information, see . . A **LayoutEventArgs** that contains the event data.

www)OnMouseDown

[C#] protected override void OnMouseDown(MouseEventArgs e);

[C++] protected: void OnMouseDown(MouseEventArgs* e);

[VB] Overrides Protected Sub OnMouseDown(ByVal e As MouseEventArgs)

[JScript] protected override function OnMouseDown(e : MouseEventArgs);

Description

Raises the

System.Windows.Forms.StatusBar.OnMouseDown(System.Windows.Forms.MouseEventArgs) event.

Raising an event invokes the event handler through a delegate. For more information, see . . A **System.Windows.Forms.MouseEventArgs** that contains the event data.

xxxx) OnMouseUp

[C#] protected override void OnMouseUp(MouseEventArgs mevent);

[C++] protected: void OnMouseUp(MouseEventArgs* mevent);

[VB] Overrides Protected Sub OnMouseUp(ByVal mevent As MouseEventArgs)

[JScript] protected override function OnMouseUp(mevent : MouseEventArgs);

Description

Raises the **System.Windows.StatusBar.OnMouseUp** event.

yyyy) OnPanelClick

[C#] protected virtual void OnPanelClick(StatusBarPanelClickEventArgs e);

[C++] protected: virtual void OnPanelClick(StatusBarPanelClickEventArgs* e);

[VB] Overridable Protected Sub OnPanelClick(ByVal e As StatusBarPanelClickEventArgs)

[JScript] protected function OnPanelClick(e : StatusBarPanelClickEventArgs);

Description

Raises the **System.Windows.Forms.StatusBar.OnPanelClick(System.Windows.Forms.StatusBarPanelClickEventArgs)** event.

Raising an event invokes the event handler through a delegate. For more information, see . A

System.Windows.Forms.StatusBarPanelClickEventArgs that contains the event data.

zzzz) OnResize

[C#] protected override void OnResize(EventArgs e);

[C++] protected: void OnResize(EventArgs* e);

[VB] Overrides Protected Sub OnResize(ByVal e As EventArgs)

[JScript] protected override function OnResize(e : EventArgs);

Description

Raises the **System.Windows.Forms.StatusBar.OnResize(System.EventArgs)** event.

Raising an event invokes the event handler through a delegate. For an overview, see . An **System.EventArgs** that contains the event data.

aaaaa) ToString

```
[C#]          public          override          string          ToString();
[C++]          public:          String*          ToString();
[VB]  Overrides  Public  Function  ToString()  As  String
[JScript]  public  override  function  ToString()  :  String;
```

Description

Returns a string representation for this control.

Return Value: String Returns a string representation for this control.

bbbbbb) WndProc

```
[C#]  protected  override  void  WndProc(ref  Message  m);
[C++]  protected:  void  WndProc(Message*  m);
[VB]  Overrides  Protected  Sub  WndProc(ByRef  m  As  Message)
[JScript]  protected  override  function  WndProc(m  :  Message);
```

Description

Base wndProc. All messages are sent to wndProc after getting filtered through the preProcessMessage function. Inheriting controls should call base.wndProc for any messages that they don't handle.

StatusBarDrawItemEventArgs class (System.Windows.Forms)

a) *WndProc*

Description

Provides data for the **System.Windows.Forms.StatusBar.DrawItem** event.

The **System.Windows.Forms.StatusBar.DrawItem** event occurs when a visual aspect of an owner-drawn **System.Windows.Forms.StatusBarPanel** changes. A **System.Windows.Forms.StatusBarDrawItemEventArgs** specifies the **System.Drawing.Graphics** object to use to draw the panel, the **System.Drawing.Rectangle** object in which to draw the panel, the panel identification number, state information about the panel, and the panel to draw. You can use the data provided by this class in an event handler for the **System.Windows.Forms.StatusBar.DrawItem** event to create custom drawn panels in your application's **System.Windows.Forms.StatusBar** control.

b) *StatusBarDrawItemEventArgs*

Example Syntax:

c) *WndProc*

```
[C#] public StatusBarDrawItemEventArgs(Graphics g, Font font, Rectangle r, int  
itemId, DrawItemState itemState, StatusBarPanel panel);
```

```
[C++] public: StatusBarDrawItemEventArgs(Graphics* g, Font* font, Rectangle r,  
int itemId, DrawItemState itemState, StatusBarPanel* panel);
```

```
[VB] Public Sub New(ByVal g As Graphics, ByVal font As Font, ByVal r As  
Rectangle, ByVal itemId As Integer, ByVal itemState As DrawItemState, ByVal  
panel As StatusBarPanel)
```

```
[JScript] public function StatusBarDrawItemEventArgs(g : Graphics, font : Font, r
```

1 : Rectangle, itemId : int, itemState : DrawItemState, panel : StatusBarPanel);

3 *Description*

4 Initializes a new instance of the
5 **System.Windows.Forms.StatusBarDrawItemEventArgs** class. The
6 **System.Drawing.Graphics** object to use to draw the
7 **System.Windows.Forms.StatusBarPanel** . The **System.Drawing.Font**
8 used to render text. The **System.Drawing.Rectangle** object that represents
9 the client area of the **System.Windows.Forms.StatusBarPanel** . The zero-
10 based index of the panel in the
11 **System.Windows.Forms.StatusBar.StatusBarPanelCollection** of the
12 **System.Windows.Forms.StatusBar** control. One of the
13 **System.Windows.Forms.DrawItemState** values that represents state
14 information about the **System.Windows.Forms.StatusBarPanel** . A
15 **System.Windows.Forms.StatusBarPanel** that represents the panel to draw.

11 d) *BackColor*

12 e) *Bounds*

13 f) *Font*

14 g) *ForeColor*

15 h) *Graphics*

16 i) *Index*

17 j) *Panel*

18 k) *WndProc*

22 *Description*

23 Gets the **System.Windows.Forms.StatusBarPanel** to draw.

24 The **System.Windows.Forms.StatusBarDrawItemEventArgs.Panel**
25 property enables you to obtain the **System.Windows.Forms.StatusBarPanel**
object that needs to be drawn. You can use this within the event handler for the

System.Windows.Forms.StatusBar.DrawItem event of a **System.Windows.Forms.StatusBar** control to perform owner-draw tasks on the **System.Windows.Forms.StatusBarPanel** that requires drawing.

l) State

StatusBarDrawItemEventHandler delegate (System.Windows.Forms)

a) ToString

Description

Represents the method that will handle the **System.Windows.Forms.StatusBar.DrawItem** event of a **System.Windows.Forms.StatusBar**. The source of the event. A **System.Windows.Forms.StatusBarDrawItemEventArgs** that contains the event data.

When you create a(n) **System.Windows.Forms.StatusBarDrawItemEventHandler** delegate, you identify the method that will handle the event. To associate the event with your event handler, add an instance of the delegate to the event. The event handler is called whenever the event occurs, unless you remove the delegate. For more information about event handler delegates, see .

StatusBarPanel class (System.Windows.Forms)

a) ToString

Description

Represents a panel in a **System.Windows.Forms.StatusBar** control.

A **System.Windows.Forms.StatusBarPanel** object represents an individual panel in the **System.Windows.Forms.StatusBar.StatusBarPanelCollection** of a **System.Windows.Forms.StatusBar** control. A **System.Windows.Forms.StatusBarPanel** object can contain text and/or an icon which can be used to reflect the status of an application. Use the **System.Windows.Forms.StatusBar.StatusBarPanelCollection**, accessible through the **System.Windows.Forms.StatusBar.Panels** property of a

System.Windows.Forms.StatusBar control, to retrieve, add, or remove an individual **System.Windows.Forms.StatusBarPanel** object.

b) StatusBarPanel

Example Syntax:

c) ToString

[C#]	public		StatusBarPanel();
[C++]	public:		StatusBarPanel();
[VB]	Public	Sub	New()
[JScript]	public	function	StatusBarPanel();

Description

Initializes a new instance of the **System.Windows.Forms.StatusBarPanel** class.

When you create an instance of **System.Windows.Forms.StatusBarPanel** , the following read/write properties are set to initial values.

d) Alignment

e) ToString

[C#]	public	HorizontalAlignment	Alignment	{get; set;}
[C++]	public:	__property	HorizontalAlignment	get_Alignment();public:
		__property	void	set_Alignment(HorizontalAlignment);
[VB]	Public	Property	Alignment	As HorizontalAlignment
[JScript]	public	function	get Alignment()	: HorizontalAlignment;public function
	set		Alignment(HorizontalAlignment);	

Description

Gets or sets the alignment of text and icons within the **System.Windows.Forms.StatusBarPanel**.

You can use this property to horizontally align the text and/or the icon assigned to the **System.Windows.Forms.StatusBarPanel.Icon** property within the borders of the panel. Text and icons can be aligned to the left, right, or center of the **System.Windows.Forms.StatusBarPanel**. There is no way to independently position an icon within the **System.Windows.Forms.StatusBarPanel**. For example, you cannot position an icon to the left side of the **System.Windows.Forms.StatusBarPanel** while aligning the text on the right side. The icon is always positioned to the left side of the text regardless of how the text is aligned.

f) *AutoSize*

g) *ToString*

```
[C#] public StatusBarPanelAutoSize AutoSize {get; set;}
```

```
[C++] public: __property StatusBarPanelAutoSize get_AutoSize();public:  
__property void set_AutoSize(StatusBarPanelAutoSize);
```

```
[VB] Public Property AutoSize As StatusBarPanelAutoSize
```

```
[JScript] public function get AutoSize() : StatusBarPanelAutoSize;public function  
set AutoSize(StatusBarPanelAutoSize);
```

Description

Gets or sets a value indicating whether the **System.Windows.Forms.StatusBarPanel** is autosized.

System.Windows.Forms.StatusBarPanel objects set to **StatusBarPanelAutoSize.Contents** have precedence over those panels set to the **StatusBarPanelAutoSize.Spring** value. For example, a **System.Windows.Forms.StatusBarPanel** that has its **System.Windows.Forms.StatusBarPanel.AutoSize** property set to **StatusBarPanelAutoSize.Spring** is shortened if a

System.Windows.Forms.StatusBarPanel object with the **System.Windows.Forms.StatusBarPanel.AutoSize** property set to **StatusBarPanelAutoSize.Contents** requires that space. You can use this property to ensure that the contents of a **System.Windows.Forms.StatusBarPanel** are properly displayed is a **System.Windows.Forms.StatusBar** control that contains more than one panel. For example, you may want a panel containing text to adjust automatically to the amount of text it is displaying (**StatusBarPanelAutoSize.Contents**), while another panel on the **System.Windows.Forms.StatusBar** that displays an owner-drawn progress bar would need to be a fixed size (**StatusBarPanelAutoSize.None**).

h) *BorderStyle*

i) *ToString*

```
[C#] public StatusBarPanelBorderStyle BorderStyle {get; set;}
```

```
[C++] public: __property StatusBarPanelBorderStyle get_BorderStyle();public:
```

```
__property void set_BorderStyle(StatusBarPanelBorderStyle);
```

```
[VB] Public Property BorderStyle As StatusBarPanelBorderStyle
```

```
[JScript] public function get BorderStyle() : StatusBarPanelBorderStyle;public
```

```
function set BorderStyle(StatusBarPanelBorderStyle);
```

Description

Gets or sets the border style of the **System.Windows.Forms.StatusBarPanel**

You can use this property to differentiate a panel from other panels in a **System.Windows.Forms.StatusBar** control.

- j) *Container*
- k) *DesignMode*
- l) *Events*
- m) *Icon*
- n) *ToString*

Description

Gets or sets the icon to display within the **System.Windows.Forms.StatusBarPanel** .

You can use this property to display an icon that represents the state of your application or a process within your application. For example, you can display an icon in a **System.Windows.Forms.StatusBarPanel** to indicate whether a file save operation is in progress or complete.

- o) *MinWidth*
- p) *ToString*

```
[C#]          public          int          MinWidth          {get;          set;}
[C++] public: __property int get_MinWidth();public: __property void
set_MinWidth(int);
[VB]          Public          Property          MinWidth          As          Integer
[JavaScript] public function get MinWidth() : int;public function set MinWidth(int);
```

Description

Gets or sets the minimum allowed width of the **System.Windows.Forms.StatusBarPanel** within the **System.Windows.Forms.StatusBar** control.

The **System.Windows.Forms.StatusBarPanel.MinWidth** property is used when the **System.Windows.Forms.StatusBarPanel.AutoSize** property is set to **StatusBarPanelAutoSize.Contents** or **StatusBarPanelAutoSize.Spring**, to prevent the **System.Windows.Forms.StatusBarPanel** from autosizing to a width that is too small. When the **System.Windows.Forms.StatusBarPanel.AutoSize** property is set to **StatusBarPanelAutoSize.None**, the **System.Windows.Forms.StatusBarPanel.MinWidth** property is automatically set to the same value as the **System.Windows.Forms.StatusBarPanel.Width** property.

q) *Parent*

r) *ToString*

[C#]	public	StatusBar	Parent	{get;}
[C++]	public:	__property	StatusBar*	get_Parent();
[VB]	Public	ReadOnly	Property	Parent As StatusBar
[JScript]	public	function	get Parent()	: StatusBar;

Description

Gets the **System.Windows.Forms.StatusBar** control that hosts the **System.Windows.Forms.StatusBarPanel**.

You can use this property to determine the **System.Windows.Forms.StatusBar** control that a **System.Windows.Forms.StatusBarPanel** object is contained within.

s) *Site*

t) *Style*

u) *ToString*

Description

Gets or sets the style of the **System.Windows.Forms.StatusBarPanel** .

You can use this property to indicate whether a **System.Windows.Forms.StatusBarPanel** displays text or whether the panel is managed as an owner-drawn **System.Windows.Forms.StatusBarPanel** . To determine when a **System.Windows.Forms.StatusBarPanel** object needs to be drawn, create an event handler for the **System.Windows.Forms.StatusBar.DrawItem** event of the **System.Windows.Forms.StatusBar** control. The **System.Windows.Forms.StatusBarDrawItemEventArgs** object passed as a parameter to an event handler of the **System.Windows.Forms.StatusBar.DrawItem** event enables you to determine which panel needs to be drawn. The **System.Windows.Forms.StatusBarDrawItemEventArgs** also provides a **System.Drawing.Graphics** object you can use to perform drawing tasks on the **System.Windows.Forms.StatusBarPanel** .

v) *Text*

w) *ToString*

```
[C#]          public          string          Text          {get;          set;}
```

```
[C++] public:  __property  String*  get_Text();public:  __property  void  
set_Text(String*);
```

```
[VB]          Public          Property          Text          As          String
```

```
[JScript] public function get Text() : String;public function set Text(String);
```

Description

Gets or sets the text of the **System.Windows.Forms.StatusBarPanel** .

This property represents the text that is displayed when the **System.Windows.Forms.StatusBarPanel.Style** property is set to **StatusBarPanelStyle.Text** . You can use this property to display information about your application in a **System.Windows.Forms.StatusBar** control. For example, you can use the **System.Windows.Forms.StatusBarPanel.Text** property to display help information when the user moves the mouse over a menu or to display the name and location of a file that is opened in an application. To align the text within **System.Windows.Forms.StatusBarPanel**

property, use the **System.Windows.Forms.StatusBarPanel.Alignment** property.

x) *ToolTipText*

y) *ToString*

```
[C#]      public      string      ToolTipText      {get;      set;}
```

```
[C++] public: __property String* get_ToolTipText();public: __property void  
set_ToolTipText(String*);
```

```
[VB]      Public      Property      ToolTipText      As      String
```

```
[JScript] public function get ToolTipText() : String;public function set  
ToolTipText(String);
```

Description

Gets or sets ToolTip text associated with the **System.Windows.Forms.StatusBarPanel**.

You can use this property to display additional information in a ToolTip when the user hovers the mouse over a **System.Windows.Forms.StatusBarPanel**. For example, you can display a ToolTip that provides data transfer speed for a **System.Windows.Forms.StatusBarPanel** that displays the status of a file transfer.

z) *Width*

aa) *ToString*

```
[C#]      public      int      Width      {get;      set;}
```

```
[C++] public: __property int get_Width();public: __property void set_Width(int);
```

```
[VB]      Public      Property      Width      As      Integer
```

```
[JScript] public function get Width() : int;public function set Width(int);
```

Description

Gets or sets the width of the **System.Windows.Forms.StatusBarPanel** within the **System.Windows.Forms.StatusBar** control.

The **System.Windows.Forms.StatusBarPanel.Width** property always reflects the actual width of a **System.Windows.Forms.StatusBarPanel** and cannot be smaller than the **System.Windows.Forms.StatusBarPanel.MinWidth** property. To autosize the width of the **System.Windows.Forms.StatusBarPanel** to the contents of the **System.Windows.Forms.StatusBarPanel** you can use the **System.Windows.Forms.StatusBarPanel.AutoSize** property.

bb) BeginInit

[C#]	public		void	BeginInit();
[C++]	public:	__sealed	void	BeginInit();
[VB]	NotOverridable	Public	Sub	BeginInit()
[JScript]	public		function	BeginInit();

Description

Begins the initialization of a **System.Windows.Forms.StatusBarPanel**.

This method is used to start the initialization of a component that is used on a form or used by another component. The **System.Windows.Forms.StatusBarPanel.EndInit** method ends the initialization. Using the **System.Windows.Forms.StatusBarPanel.BeginInit** and **System.Windows.Forms.StatusBarPanel.EndInit** methods prevents the control from being used before it is fully initialized.

cc) Dispose

[C#]	protected	override	void	Dispose(bool disposing);
[C++]	protected:		void	Dispose(bool disposing);

1 [VB] Overrides Protected Sub Dispose(ByVal disposing As Boolean)
 2 [JScript] protected override function Dispose(disposing : Boolean);

4 *Description*

5 [To be supplied.]

6 *dd) EndInit*

8 [C#] public void EndInit();
 9 [C++] public: __sealed void EndInit();
 10 [VB] NotOverridable Public Sub EndInit()
 11 [JScript] public function EndInit();

13 *Description*

14 Ends the initialization of a **System.Windows.Forms.StatusBarPanel** .

15 This method is used to end the initialization of a component that is used by
 16 another component. The

17 **System.Windows.Forms.StatusBarPanel.BeginInit** method starts the
 18 initialization. Using the **System.Windows.Forms.StatusBarPanel.BeginInit**
 19 and **System.Windows.Forms.StatusBarPanel.EndInit** methods prevents
 20 the control from being used before it is fully initialized.

19 *ee) ToString*

21 [C#] public override string ToString();
 22 [C++] public: String* ToString();
 23 [VB] Overrides Public Function ToString() As String
 24 [JScript] public override function ToString() : String;

1
2 *Description*

3 Retrieves a string that contains information about the panel.

4 *Return Value:* Returns a string that contains the class name for the control and the text it contains.

5 StatusBarPanelAutoSize enumeration (System.Windows.Forms)

6 *a) ToString*

7
8
9 *Description*

10 Specifies how a **System.Windows.Forms.StatusBarPanel** on a
11 **System.Windows.Forms.StatusBar** control autosizes when the control
resizes.

12 This enumeration is used by the
13 **System.Windows.Forms.StatusBarPanel.AutoSize** property. The
14 **System.Windows.Forms.StatusBarPanel.AutoSize** property specifies how a
15 **System.Windows.Forms.StatusBarPanel** is autosized on a
16 **System.Windows.Forms.StatusBar** control.

17
18 *b) ToString*

19 [C#] public const StatusBarPanelAutoSize Contents;

20 [C++] public: const StatusBarPanelAutoSize Contents;

21 [VB] Public Const Contents As StatusBarPanelAutoSize

22 [JScript] public var Contents : StatusBarPanelAutoSize;

23
24 *Description*

25 The width of the StatusBarPanel is determined by its contents.

c) *ToString*

[C#]	public	const	StatusBarPanelAutoSize		None;
[C++]	public:	const	StatusBarPanelAutoSize		None;
[VB]	Public	Const	None	As	StatusBarPanelAutoSize
[JScript]	public	var	None	:	StatusBarPanelAutoSize;

Description

The **System.Windows.Forms.StatusBarPanel** does not change size when the **System.Windows.Forms.StatusBar** control resizes.

d) *ToString*

[C#]	public	const	StatusBarPanelAutoSize		Spring;
[C++]	public:	const	StatusBarPanelAutoSize		Spring;
[VB]	Public	Const	Spring	As	StatusBarPanelAutoSize
[JScript]	public	var	Spring	:	StatusBarPanelAutoSize;

Description

The **System.Windows.Forms.StatusBarPanel** shares the the available space on the **System.Windows.Forms.StatusBar** (the space not taken up by other panels whose **System.Windows.Forms.StatusBarPanel.AutoSize** property is set to **StatusBarPanelAutoSize.None** or **StatusBarPanelAutoSize.Contents**) with other panels that have their **System.Windows.Forms.StatusBarPanel.AutoSize** property set to **StatusBarPanelAutoSize.Spring** .

StatusBarPanelBorderStyle enumeration (System.Windows.Forms)

a) ToString

Description

Specifies the appearance of the border for a **System.Windows.Forms.StatusBarPanel** on a **System.Windows.Forms.StatusBar** control.

This enumeration is used by the **System.Windows.Forms.StatusBarPanel.BorderStyle** property of the **System.Windows.Forms.StatusBarPanel** class. The **System.Windows.Forms.StatusBarPanel.BorderStyle** property enables you to specify the style of border to display a **System.Windows.Forms.StatusBarPanel** within a **System.Windows.Forms.StatusBar**.

b) ToString

[C#]	public	const	StatusBarPanelBorderStyle	None;
[C++]	public:	const	StatusBarPanelBorderStyle	None;
[VB]	Public	Const	None	As StatusBarPanelBorderStyle
[JScript]	public	var	None	: StatusBarPanelBorderStyle;

Description

No border is displayed.

c) ToString

[C#]	public	const	StatusBarPanelBorderStyle	Raised;
[C++]	public:	const	StatusBarPanelBorderStyle	Raised;

```

1 [VB]      Public      Const      Raised      As      StatusBarPanelBorderStyle
2 [JScript]    public      var      Raised      :      StatusBarPanelBorderStyle;

```

Description

The **System.Windows.Forms.StatusBarPanel** is displayed with a three-dimensional raised border.

d) ToString

```

8 [C#]      public      const      StatusBarPanelBorderStyle      Sunken;
9 [C++]      public:      const      StatusBarPanelBorderStyle      Sunken;
10 [VB]      Public      Const      Sunken      As      StatusBarPanelBorderStyle
11 [JScript]    public      var      Sunken      :      StatusBarPanelBorderStyle;

```

Description

The **System.Windows.Forms.StatusBarPanel** is displayed with a three-dimensional sunken border.

StatusBarPanelClickEventArgs class (System.Windows.Forms)

a) ToString

Description

Provides data for the **System.Windows.Forms.StatusBar.PanelClick** event.

The **System.Windows.Forms.StatusBar.PanelClick** event occurs when the user clicks a panel on the **System.Windows.Forms.StatusBar** . A

System.Windows.Forms.StatusBarPanelClickEventArgs specifies which **System.Windows.Forms.StatusBarPanel** was clicked, the mouse button that was pressed, how many times it was pressed, and the coordinates of the mouse click at the time the **System.Windows.Forms.StatusBarPanel** was clicked.

You can use the data provided by this class in an event handler for the **System.Windows.Forms.StatusBar.PanelClick** event to perform tasks related to the **System.Windows.Forms.StatusBarPanel** being clicked. For example, if a **System.Windows.Forms.StatusBarPanel** is used to display the time, you could create an event handler for the **System.Windows.Forms.StatusBar.PanelClick** event and use data provided by this class to display a dialog that allows you to modify the date and time on the system.

b) StatusBarPanelClickEventArgs

Example Syntax:

c) ToString

```
[C#] public StatusBarPanelClickEventArgs(StatusBarPanel statusBarPanel,
MouseButtons button, int clicks, int x, int y);

[C++] public: StatusBarPanelClickEventArgs(StatusBarPanel* statusBarPanel,
MouseButtons button, int clicks, int x, int y);

[VB] Public Sub New(ByVal statusBarPanel As StatusBarPanel, ByVal button As
MouseButtons, ByVal clicks As Integer, ByVal x As Integer, ByVal y As Integer)

[JScript] public function StatusBarPanelClickEventArgs(statusBarPanel :
StatusBarPanel, button : MouseButtons, clicks : int, x : int, y : int);
```

Description

Initializes a new instance of the **System.Windows.Forms.StatusBarPanelClickEventArgs** class. The **System.Windows.Forms.StatusBarPanel** that represents the panel that was clicked. One of the **System.Windows.Forms.MouseButtons** values that represents the mouse buttons that were clicked while over the **System.Windows.Forms.StatusBarPanel**. The number of times that the mouse button was clicked. The x-coordinate of the mouse click. The y-coordinate of the mouse click.

1 d) *Button*

2 e) *Clicks*

3 f) *Delta*

4 g) *StatusBarPanel*

5 h) *ToString*

6
7
8 *Description*

9 Gets the **System.Windows.Forms.StatusBarPanel** to draw.

10 The
11 **System.Windows.Forms.StatusBarPanelClickEventArgs.StatusBarPanel**
12 property enables you to obtain the **System.Windows.Forms.StatusBarPanel**
13 object that was clicked. You can use this within the event handler for the
14 **System.Windows.Forms.StatusBar.PanelClick** event of a
15 **System.Windows.Forms.StatusBar** control to perform tasks such as
16 displaying custom dialog boxes when a
17 **System.Windows.Forms.StatusBarPanel** within a
18 **System.Windows.Forms.StatusBar** control is clicked.

15 i) *X*

16 j) *Y*

17 StatusBarPanelClickEventHandler delegate (System.Windows.Forms)

18
19 a) *ToString*

20
21
22 *Description*

23 Represents the method that will handle the
24 **System.Windows.Forms.StatusBar.PanelClick** event of a
25 **System.Windows.Forms.StatusBar** . The source of the event. A
26 **System.Windows.Forms.StatusBarPanelClickEventArgs** that contains the
27 event data.

When you create a(n)

System.Windows.Forms.StatusBarPanelClickEventHandler delegate, you identify the method that will handle the event. To associate the event with your event handler, add an instance of the delegate to the event. The event handler is called whenever the event occurs, unless you remove the delegate. For more information about event handler delegates, see .

StatusBar.StatusBarPanelCollection class (System.Windows.Forms)

a) *ToString*

Description

Represents the collection of panels in a **System.Windows.Forms.StatusBar** control.

The **System.Windows.Forms.StatusBar.StatusBarPanelCollection** class stores the panels displayed in the **System.Windows.Forms.StatusBar** . Each object in the collection is an instance of the StatusBarPanel class which defines the display characteristics and behaviors of a panel displayed in a **System.Windows.Forms.StatusBar** .

b) *StatusBar.StatusBarPanelCollection*

Example Syntax:

c) *ToString*

[C#] public StatusBar.StatusBarPanelCollection(StatusBar owner);

[C++] public: StatusBarPanelCollection(StatusBar* owner);

[VB] Public Sub New(ByVal owner As StatusBar)

[JScript] public function StatusBar.StatusBarPanelCollection(owner : StatusBar);

Description

Initializes a new instance of the

System.Windows.Forms.StatusBar.StatusBarPanelCollection class.

You cannot create an instance of this class without associating it with a **System.Windows.Forms.StatusBar** control. The **System.Windows.Forms.StatusBar** control that contains this collection.

d) *Count*

e) *ToString*

[C#]	public	int	Count	{get;}
[C++]	public:	__property	int	get_Count();
[VB]	Public	ReadOnly	Property	Count As Integer
[JScript]	public	function	get	Count() : int;

Description

Gets the number of items in the collection.

This property enables you to determine the number of panels in the **System.Windows.Forms.StatusBar**. You can then use this value when you are looping through the values of the collection and you need to provide a number of iterations to perform the loop.

f) *IsReadOnly*

g) *ToString*

[C#]	public	bool	IsReadOnly	{get;}
[C++]	public:	__property	bool	get_IsReadOnly();
[VB]	Public	ReadOnly	Property	IsReadOnly As Boolean
[JScript]	public	function	get	IsReadOnly() : Boolean;

Description

Gets a value indicating whether this collection is read-only.

This property is always **false** for this collection.

h) Item

i) ToString

[C#] public virtual StatusBarPanel this[int index] {get; set;}

[C++] public: __property virtual StatusBarPanel* get_Item(int index);public:

__property virtual void set_Item(int index, StatusBarPanel*);

[VB] Overridable Public Default Property Item(ByVal index As Integer) As

StatusBarPanel

[JScript] returnValue =

StatusBarPanelCollectionObject.Item(index);StatusBarPanelCollectionObject.Item

(index) = returnValue;

Description

Gets or sets the **System.Windows.Forms.StatusBarPanel** at the specified index.

You can use this method to obtain the

System.Windows.Forms.StatusBarPanel stored at a specific location within the collection. To determine the index of a specific panel within the collection, use the

System.Windows.Forms.StatusBar.StatusBarPanelCollection.IndexOf(System.Windows.Forms.StatusBarPanel) method. The index of the panel in the collection to get or set.

j) Add

[C#] public virtual int Add(StatusBarPanel value);

[C++] public: virtual int Add(StatusBarPanel* value);

[VB] Overridable Public Function Add(ByVal value As StatusBarPanel) As

Integer

```
[JScript] public function Add(value : StatusBarPanel) : int;
```

Description

Adds a **System.Windows.Forms.StatusBarPanel** to the collection.

Return Value: The zero-based index of the item in the collection.

You can add panels to a **System.Windows.Forms.StatusBar** control to display more than one type of information. This version of the **System.Windows.Forms.StatusBar.StatusBarPanelCollection.Add(System.String)** method adds the **System.Windows.Forms.StatusBarPanel** object specified in the *value* parameter to the collection. The order in which panels are located in the **System.Windows.Forms.StatusBar.StatusBarPanelCollection** represents the order that panels are displayed within the **System.Windows.Forms.StatusBar** control. Panels are displayed from left to right starting with the first panel in the collection. The **System.Windows.Forms.Control.RightToLeft** property of the **System.Windows.Forms.StatusBar** control does not change the order in which panels are displayed in the **System.Windows.Forms.StatusBar**. To insert a panel at a specific position in the collection, use the **System.Windows.Forms.StatusBar.StatusBarPanelCollection.Insert(System.Int32, System.Windows.Forms.StatusBarPanel)** method. To add a set of panels to the collection in a single operation, use the **System.Windows.Forms.StatusBar.StatusBarPanelCollection.AddRange(System.Windows.Forms.StatusBarPanel[])** method. A **System.Windows.Forms.StatusBarPanel** that represents the panel to add to the collection.

k) Add

```
[C#] public virtual StatusBarPanel Add(string text);
```

```
[C++] public: virtual StatusBarPanel* Add(String* text);
```

```
[VB] Overridable Public Function Add(ByVal text As String) As StatusBarPanel
```

```
[JScript] public function Add(text : String) : StatusBarPanel; Adds a  
System.Windows.Forms.StatusBarPanel to the collection.
```

Description

Adds a **System.Windows.Forms.StatusBarPanel** with the specified text to the collection.

Return Value: A **System.Windows.Forms.StatusBarPanel** that represents the panel that was added to the collection.

You can add panels to a **System.Windows.Forms.StatusBar** control to display more than one type of information. This version of the **System.Windows.Forms.StatusBar.StatusBarPanelCollection.Add(System.String)** method creates a new **System.Windows.Forms.StatusBarPanel** object with the text specified in the *text* parameter and adds it to collection. The order in which panels are located in the **System.Windows.Forms.StatusBar.StatusBarPanelCollection** represents the order that panels are displayed within the **System.Windows.Forms.StatusBar** control. Panels are displayed from left to right starting with the first panel in the collection. The **System.Windows.Forms.Control.RightToLeft** property of the **System.Windows.Forms.StatusBar** control does not change the order in which panels are displayed in the **System.Windows.Forms.StatusBar**. To insert a panel at a specific position in the collection, use the **System.Windows.Forms.StatusBar.StatusBarPanelCollection.Insert(System.Int32, System.Windows.Forms.StatusBarPanel)** method. To add a set of panels to the collection in a single operation, use the **System.Windows.Forms.StatusBar.StatusBarPanelCollection.AddRange(System.Windows.Forms.StatusBarPanel[])** method. The text for the **System.Windows.Forms.StatusBarPanel** that is being added.

1) AddRange

```
[C#]    public    virtual    void    AddRange(StatusBarPanel[]    panels);
[C++]    public:    virtual    void    AddRange(StatusBarPanel*    panels[]);
[VB]    Overridable Public Sub AddRange(ByVal panels() As StatusBarPanel)
[JScript]    public    function    AddRange(panels    :    StatusBarPanel[]);
```

Description

Adds an array of **System.Windows.Forms.StatusBarPanel** objects to the collection.

You can add panels to a **System.Windows.Forms.StatusBar** control to display more than one type of information. The **System.Windows.Forms.StatusBar.StatusBarPanelCollection.AddRange(System.Windows.Forms.StatusBarPanel[])** method adds an array of **System.Windows.Forms.StatusBarPanel** objects specified in the *panels* parameter to the collection. All existing panels in the collection are removed from the collection before inserting new panels. The order in which panels are located in the **System.Windows.Forms.StatusBar.StatusBarPanelCollection** represents the order that panels are displayed within the **System.Windows.Forms.StatusBar** control. Panels are displayed from left to right starting with the first panel in the collection. The **System.Windows.Forms.Control.RightToLeft** property of the **System.Windows.Forms.StatusBar** control does not change the order in which panels are displayed in the **System.Windows.Forms.StatusBar**. To add a single **System.Windows.Forms.StatusBarPanel** object to the collection, use the **System.Windows.Forms.StatusBar.StatusBarPanelCollection.Add(System.String)** method. To insert a panel at a specific position in the **System.Windows.Forms.StatusBar** control (and in this collection), use the **System.Windows.Forms.StatusBar.StatusBarPanelCollection.Insert(System.Int32, System.Windows.Forms.StatusBarPanel)** method. An array of **System.Windows.Forms.StatusBarPanel** objects to add.

m) Clear

[C#]	public	virtual	void	Clear();
[C++]	public:	virtual	void	Clear();
[VB]	Overridable	Public	Sub	Clear()
[JScript]	public	function		Clear();

Description

Removes all items from the collection.

When you remove panels from the collection, all information about the deleted panels is lost. To remove a single panel from the **System.Windows.Forms.StatusBar**, use the

System.Windows.Forms.StatusBar.StatusBarPanelCollection.Remove(System.Windows.Forms.StatusBarPanel) or
System.Windows.Forms.StatusBar.StatusBarPanelCollection.RemoveAt(System.Int32) method.

n) Contains

[C#] public bool Contains(StatusBarPanel panel);

[C++] public: bool Contains(StatusBarPanel* panel);

[VB] Public Function Contains(ByVal panel As StatusBarPanel) As Boolean

[JScript] public function Contains(panel : StatusBarPanel) : Boolean;

Description

Determines whether the specified panel is located within the collection.

Return Value: **true** if the panel is located within the collection; otherwise, **false** .

The

System.Windows.Forms.StatusBar.StatusBarPanelCollection.Contains(System.Windows.Forms.StatusBarPanel) method enables you to determine whether a panel is a member of the collection. Once you know that the item is located within the collection, you can use the

System.Windows.Forms.StatusBar.StatusBarPanelCollection.IndexOf(System.Windows.Forms.StatusBarPanel) method to determine where the panel is located within the collection. The **System.Windows.Forms.StatusBarPanel** to locate in the collection.

o) GetEnumerator

[C#] public IEnumerator GetEnumerator();

[C++] public: __sealed IEnumerator* GetEnumerator();

[VB] NotOverridable Public Function GetEnumerator() As IEnumerator

[JScript] public function GetEnumerator() : IEnumerator;

Description

Returns an enumerator to use to iterate through the item collection.

Return Value: An **System.Collections.IEnumerator** object that represents the item collection.

p) IndexOf

[C#] public int IndexOf(StatusBarPanel panel);

[C++] public: int IndexOf(StatusBarPanel* panel);

[VB] Public Function IndexOf(ByVal panel As StatusBarPanel) As Integer

[JScript] public function IndexOf(panel : StatusBarPanel) : int;

Description

Returns the index within the collection of the specified panel.

Return Value: The zero-based index where the panel is located within the collection; otherwise, negative one (-1).

The

System.Windows.Forms.StatusBar.StatusBarPanelCollection.IndexOf(System.Windows.Forms.StatusBarPanel) method enables you to determine where a panel is located within the collection. To determine whether an item is located within the collection before calling this method, use the **System.Windows.Forms.StatusBar.StatusBarPanelCollection.Contains(System.Windows.Forms.StatusBarPanel)** method. The **System.Windows.Forms.StatusBarPanel** object to locate in the collection.

q) Insert

[C#] public virtual void Insert(int index, StatusBarPanel value);

[C++] public: virtual void Insert(int index, StatusBarPanel* value);

[VB] Overridable Public Sub Insert(ByVal index As Integer, ByVal value As StatusBarPanel)

[JScript] public function Insert(index : int, value : StatusBarPanel);

Description

Inserts the specified **System.Windows.Forms.StatusBarPanel** into the collection at the specified index.

You can add panels to a **System.Windows.Forms.StatusBar** control to display more than one type of information. The **System.Windows.Forms.StatusBar.StatusBarPanelCollection.Insert(System.Int32, System.Windows.Forms.StatusBarPanel)** method enables you to create a new **System.Windows.Forms.StatusBarPanel** object and insert it at a specific location within the collection. The order in which panels are located in the **System.Windows.Forms.StatusBar.StatusBarPanelCollection** represents the order that panels are displayed within the **System.Windows.Forms.StatusBar** control. Panels are displayed from left to right starting with the first panel in the collection. The **System.Windows.Forms.Control.RightToLeft** property of the **System.Windows.Forms.StatusBar** control does not change the order in which panels are displayed in the **System.Windows.Forms.StatusBar**. To add a panel to the end of the collection, use the **System.Windows.Forms.StatusBar.StatusBarPanelCollection.Add(System.String)** method. To add a set of panels to the collection in a single operation, use the **System.Windows.Forms.StatusBar.StatusBarPanelCollection.AddRange(System.Windows.Forms.StatusBarPanel[])** method. The zero-based index location where the panel is inserted. A **System.Windows.Forms.StatusBarPanel** representing the panel to insert.

r) Remove

[C#] public virtual void Remove(StatusBarPanel value);

[C++] public: virtual void Remove(StatusBarPanel* value);

[VB] Overridable Public Sub Remove(ByVal value As StatusBarPanel)

[JScript] public function Remove(value : StatusBarPanel);

Description

Removes the specified **System.Windows.Forms.StatusBarPanel** from the collection.

When you remove a panel from the list, the indices change for subsequent items in the collection. All information about the removed panel is deleted. You can use this method to remove a specific panel from the

System.Windows.Forms.StatusBar by specifying the actual **System.Windows.Forms.StatusBarPanel** to remove from the collection. To specify the index of the panel to remove instead of the panel itself, use the **System.Windows.Forms.StatusBar.StatusBarPanelCollection.RemoveAt(System.Int32)** method. To remove all panels from the collection, use the **System.Windows.Forms.StatusBar.StatusBarPanelCollection.Clear** method. The **System.Windows.Forms.StatusBarPanel** representing the panel to remove from the collection.

s) *RemoveAt*

[C#] public virtual void RemoveAt(int index);

[C++] public: virtual void RemoveAt(int index);

[VB] Overridable Public Sub RemoveAt(ByVal index As Integer)

[JScript] public function RemoveAt(index : int);

Description

Removes the **System.Windows.Forms.StatusBarPanel** located at the specified index within the collection.

When you remove a panel from the collection, the indices change for subsequent panels in the collection. All information about the removed panel is deleted. You can use this method to remove a specific panel from the list by specifying the index of the panel to remove from the collection. To specify the panel to remove instead of the index to the panel, use the

System.Windows.Forms.StatusBar.StatusBarPanelCollection.Remove(System.Windows.Forms.StatusBarPanel) method. To remove all panels from the **System.Windows.Forms.StatusBar** control, use the **System.Windows.Forms.StatusBar.StatusBarPanelCollection.Clear** method. The zero-based index of the item to remove.

t) *ICollection.CopyTo*

[C#] void ICollection.CopyTo(Array dest, int index);

[C++] void ICollection::CopyTo(Array* dest, int index);

[VB] Sub CopyTo(ByVal dest As Array, ByVal index As Integer) Implements
ICollection.CopyTo

[JScript] function ICollection.CopyTo(dest : Array, index : int);

u) *IList.Add*

[C#] int IList.Add(object value);

[C++] int IList::Add(Object* value);

[VB] Function Add(ByVal value As Object) As Integer Implements IList.Add

[JScript] function IList.Add(value : Object) : int;

v) *IList.Contains*

[C#] bool IList.Contains(object panel);

[C++] bool IList::Contains(Object* panel);

[VB] Function Contains(ByVal panel As Object) As Boolean Implements
IList.Contains

[JScript] function IList.Contains(panel : Object) : Boolean;

w) *IList.IndexOf*

[C#] int IList.IndexOf(object panel);

[C++] int IList::IndexOf(Object* panel);

[VB] Function IndexOf(ByVal panel As Object) As Integer Implements

1 IList.IndexOf

2 [JScript] function IList.IndexOf(panel : Object) : int;

3 x) *IList.Insert*

5 [C#] void IList.Insert(int index, object value);

6 [C++] void IList::Insert(int index, Object* value);

7 [VB] Sub Insert(ByVal index As Integer, ByVal value As Object) Implements
8 IList.Insert

9 [JScript] function IList.Insert(index : int, value : Object);

10 y) *IList.Remove*

12 [C#] void IList.Remove(object value);

13 [C++] void IList::Remove(Object* value);

14 [VB] Sub Remove(ByVal value As Object) Implements IList.Remove

15 [JScript] function IList.Remove(value : Object);

16 StatusBarPanelStyle enumeration (System.Windows.Forms)

17 a) *ToString*

20 *Description*

21 Specifies whether a **System.Windows.Forms.StatusBarPanel** object on a
22 **System.Windows.Forms.StatusBar** control is owner-drawn or system-drawn.

23 Use the members of this enumeration to set the value of the
24 **System.Windows.Forms.StatusBarPanel.Style** property of the
25 **System.Windows.Forms.StatusBarPanel** class. The Style property
determines the way in which the **System.Windows.Forms.StatusBarPanel**
will be displayed.

b) ToString

```
[C#]      public      const      StatusBarPanelStyle      OwnerDraw;
[C++]     public:     const      StatusBarPanelStyle      OwnerDraw;
[VB]      Public      Const      OwnerDraw      As      StatusBarPanelStyle
[JScript] public      var      OwnerDraw      :      StatusBarPanelStyle;
```

Description

The **System.Windows.Forms.StatusBarPanel** is drawn by the owner.

c) ToString

```
[C#]      public      const      StatusBarPanelStyle      Text;
[C++]     public:     const      StatusBarPanelStyle      Text;
[VB]      Public      Const      Text      As      StatusBarPanelStyle
[JScript] public      var      Text      :      StatusBarPanelStyle;
```

Description

The **System.Windows.Forms.StatusBarPanel** displays text in the standard font.

StructFormat enumeration (System.Windows.Forms)

a) ToString

Description

b) ToString

[C#]	public	const	StructFormat	Ansi;
[C++]	public:	const	StructFormat	Ansi;
[VB]	Public	Const	Ansi As	StructFormat
[JScript]	public	var	Ansi :	StructFormat;

Description

c) ToString

[C#]	public	const	StructFormat	Auto;
[C++]	public:	const	StructFormat	Auto;
[VB]	Public	Const	Auto As	StructFormat
[JScript]	public	var	Auto :	StructFormat;

Description

d) ToString

[C#]	public	const	StructFormat	Unicode;
[C++]	public:	const	StructFormat	Unicode;
[VB]	Public	Const	Unicode As	StructFormat
[JScript]	public	var	Unicode :	StructFormat;

1
2 *Description*

3
4 SystemInformation class (System.Windows.Forms)

5 a) *ToString*

6
7
8 *Description*

9 Provides information about the operating system.

10 **System.Windows.Forms.SystemInformation** provides **static** methods and
11 properties that can be used to get information such as Windows display element
12 sizes, operating system settings, network availability, and the capabilities of
13 hardware installed on the system. This class cannot be instantiated.

14 b) *ArrangeDirection*

15 c) *ToString*

16 [C#] public static ArrangeDirection ArrangeDirection {get;}

17 [C++] public: __property static ArrangeDirection get_ArrangeDirection();

18 [VB] Public Shared ReadOnly Property ArrangeDirection As ArrangeDirection

19 [JScript] public static function get ArrangeDirection() : ArrangeDirection;

20
21 *Description*

22 Gets flags specifying how the operating system arranges minimized windows.
23
24
25

d) *ArrangeStartingPosition*

e) *ToString*

[C#] public static ArrangeStartingPosition ArrangeStartingPosition {get;}

[C++] public: __property static ArrangeStartingPosition
get_ArrangeStartingPosition();

[VB] Public Shared ReadOnly Property ArrangeStartingPosition As
ArrangeStartingPosition

[JScript] public static function get ArrangeStartingPosition() :
ArrangeStartingPosition;

Description

Gets flags specifying how the operating system arranges minimized windows.

The value is a combination of one starting position value and one direction value.

f) *BootMode*

g) *ToString*

[C#] public static BootMode BootMode {get;}

[C++] public: __property static BootMode get_BootMode();

[VB] Public Shared ReadOnly Property BootMode As BootMode

[JScript] public static function get BootMode() : BootMode;

Description

Gets a value that specifies how the system was started.

Use **System.Windows.Forms.SystemInformation.BootMode** to determine how the user started the system. For applications running on Windows 95 or

Windows 98, you can use this property to determine whether the operating system is in fail-safe mode. Your application uses this information to operate appropriately when it accesses system services and hardware.

h) Border3DSize

i) ToString

```
[C#]      public      static      Size      Border3DSize      {get;}
```

```
[C++]     public:     __property static Size get_Border3DSize();
```

```
[VB]      Public      Shared      ReadOnly Property Border3DSize As Size
```

```
[JScript] public static function get Border3DSize() : Size;
```

Description

Gets the dimensions, in pixels, of a three-dimensional (3-D) border.

System.Windows.Forms.SystemInformation.Border3DSize is the three-dimensional counterpart of the **System.Windows.Forms.SystemInformation.BorderSize**. Use this property when creating a control that has a 3-D border to determine the proper sizing of the border.

j) BorderSize

k) ToString

```
[C#]      public      static      Size      BorderSize      {get;}
```

```
[C++]     public:     __property static Size get_BorderSize();
```

```
[VB]      Public      Shared      ReadOnly Property BorderSize As Size
```

```
[JScript] public static function get BorderSize() : Size;
```

Description

Gets the width and height, in pixels, of a window border.

The **System.Windows.Forms.SystemInformation.BorderSize** property is the non-three-dimensional counterpart of the **System.Windows.Forms.SystemInformation.Border3DSize** property. Use this property when creating a control that has a border to determine the proper sizing of the border.

l) *CaptionButtonSize*

m) *ToString*

[C#] public static Size CaptionButtonSize {get;}

[C++] public: __property static Size get_CaptionButtonSize();

[VB] Public Shared ReadOnly Property CaptionButtonSize As Size

[JScript] public static function get CaptionButtonSize() : Size;

Description

Gets the dimensions, in pixels, of a caption bar or title bar button.

Use **System.Windows.Forms.SystemInformation.CaptionButtonSize** to get the dimensions of a button in the caption bar of window, to place additional caption buttons in the caption of a window, and to ensure the button is properly sized.

n) *CaptionHeight*

o) *ToString*

[C#] public static int CaptionHeight {get;}

[C++] public: __property static int get_CaptionHeight();

[VB] Public Shared ReadOnly Property CaptionHeight As Integer

[JScript] public static function get CaptionHeight() : int;

Description

Gets the height, in pixels, of the normal caption area of a window.

Use **System.Windows.Forms.SystemInformation.CaptionHeight** to determine the size of the standard window caption. You can also perform special display operations or add a caption button to the caption of a window.

p) ComputerName

q) ToString

```
[C#]      public      static      string      ComputerName      {get;}
```

```
[C++]     public:     __property     static     String*     get_ComputerName();
```

```
[VB]      Public      Shared      ReadOnly      Property      ComputerName      As      String
```

```
[JScript] public      static      function      get      ComputerName()      :      String;
```

Description

Gets the computer name of the current system.

Use **System.Windows.Forms.SystemInformation.ComputerName** to determine the name of the computer as it is displayed to other users on a network. This name is established at system startup, when it is initialized from the Registry.

r) CursorSize

s) ToString

```
[C#]      public      static      Size      CursorSize      {get;}
```

```
[C++]     public:     __property     static      Size      get_CursorSize();
```

```
[VB]      Public      Shared      ReadOnly      Property      CursorSize      As      Size
```

```
[JScript] public      static      function      get      CursorSize()      :      Size;
```

Description

Gets the dimensions, in pixels, of a cursor.

The system cannot create cursors of other sizes.

t) *DbcsEnabled*

u) *ToString*

```
[C#]      public      static      bool      DbcsEnabled      {get;}
```

```
[C++]     public:     __property     static     bool     get_DbcsEnabled();
```

```
[VB]      Public      Shared      ReadOnly      Property      DbcsEnabled      As      Boolean
```

```
[JScript] public      static      function      get      DbcsEnabled()      :      Boolean;
```

Description

Gets a value indicating whether the operating system is capable of handling double-byte character set (DBCS) characters.

Use **System.Windows.Forms.SystemInformation.DbcsEnabled** to determine whether the operating system on which your application is running supports DBCS. Although an operating system can support DBCS, this does not mean that the user runs a culture that uses DBCS.

v) *DebugOS*

w) *ToString*

```
[C#]      public      static      bool      DebugOS      {get;}
```

```
[C++]     public:     __property     static     bool     get_DebugOS();
```

```
[VB]      Public      Shared      ReadOnly      Property      DebugOS      As      Boolean
```

```
[JScript] public      static      function      get      DebugOS()      :      Boolean;
```

Description

Gets a value indicating whether this is a debug version of the operating system.

x) *DoubleClickSize*

y) *ToString*

[C#] public static Size DoubleClickSize {get;}

[C++] public: __property static Size get_DoubleClickSize();

[VB] Public Shared ReadOnly Property DoubleClickSize As Size

[JScript] public static function get DoubleClickSize() : Size;

Description

Gets the dimensions, in pixels, of the area within which the user must click for the operating system to consider the two clicks a double-click.

The rectangle is centered around the first click.

z) *DoubleClickTime*

aa) *ToString*

[C#] public static int DoubleClickTime {get;}

[C++] public: __property static int get_DoubleClickTime();

[VB] Public Shared ReadOnly Property DoubleClickTime As Integer

[JScript] public static function get DoubleClickTime() : int;

Description

Gets the maximum number of milliseconds allowed between mouse clicks for a double-click to be valid.

A double-click is a series of two clicks of the mouse button, the second occurring within a specified length of time after the first. The double-click time is the maximum number of milliseconds that can occur between the first and second click of a double-click. For a double-click to be registered, it must also be within

the same region. To determine this region, use

System.Windows.Forms.SystemInformation.DoubleClickSize .

bb) DragFullWindows

cc) ToString

[C#] public static bool DragFullWindows {get;}

[C++] public: __property static bool get_DragFullWindows();

[VB] Public Shared ReadOnly Property DragFullWindows As Boolean

[JScript] public static function get DragFullWindows() : Boolean;

Description

Gets a value indicating whether the user has enabled full window drag.

When this property is **true** , the contents of windows are displayed when moving and sizing.

dd) DragSize

ee) ToString

[C#] public static Size DragSize {get;}

[C++] public: __property static Size get_DragSize();

[VB] Public Shared ReadOnly Property DragSize As Size

[JScript] public static function get DragSize() : Size;

Description

Gets the dimensions, in pixels, of the rectangle that a drag operation must extend to be considered a drag operation. The rectangle is centered on a drag point.

Use **System.Windows.Forms.SystemInformation.DragSize** to determine the size of the rectangle that Windows uses as a boundary before starting a drag operation. This rectangle allows limited movement before a drag operation begins, enabling the user to click and release the mouse button easily without unintentionally starting a drag operation.

ff) FixedFrameBorderSize

gg) ToString

[C#] public static Size FixedFrameBorderSize {get;}

[C++] public: __property static Size get_FixedFrameBorderSize();

[VB] Public Shared ReadOnly Property FixedFrameBorderSize As Size

[JScript] public static function get FixedFrameBorderSize() : Size;

Description

Gets the thickness, in pixels, of the border for a window that has a caption and is not resizable.

hh) FrameBorderSize

ii) ToString

[C#] public static Size FrameBorderSize {get;}

[C++] public: __property static Size get_FrameBorderSize();

[VB] Public Shared ReadOnly Property FrameBorderSize As Size

[JScript] public static function get FrameBorderSize() : Size;

Description

Gets the thickness, in pixels, of the border for a window that can be resized.

1 *jj) HighContrast*

2 *kk) ToString*

3
4 [C#] public static bool HighContrast {get;}

5 [C++] public: __property static bool get_HighContrast();

6 [VB] Public Shared ReadOnly Property HighContrast As Boolean

7 [JScript] public static function get HighContrast() : Boolean;

8
9 *Description*

10 Gets a value indicating whether the user has selected to run in high-contrast
11 mode.

12 *ll) HorizontalScrollBarArrowWidth*

13 *mm) ToString*

14 [C#] public static int HorizontalScrollBarArrowWidth {get;}

15 [C++] public: __property static int get_HorizontalScrollBarArrowWidth();

16 [VB] Public Shared ReadOnly Property HorizontalScrollBarArrowWidth As
17 Integer

18 [JScript] public static function get HorizontalScrollBarArrowWidth() : int;

19
20 *Description*

21 Gets the width, in pixels, of the arrow bitmap on the horizontal scroll bar.
22
23
24
25

nn) *HorizontalScrollBarHeight*

oo) *ToString*

[C#] public static int HorizontalScrollBarHeight {get;}

[C++] public: __property static int get_HorizontalScrollBarHeight();

[VB] Public Shared ReadOnly Property HorizontalScrollBarHeight As Integer

[JScript] public static function get HorizontalScrollBarHeight() : int;

Description

Gets the height, in pixels, of the horizontal scroll bar.

pp) *HorizontalScrollBarThumbWidth*

qq) *ToString*

[C#] public static int HorizontalScrollBarThumbWidth {get;}

[C++] public: __property static int get_HorizontalScrollBarThumbWidth();

[VB] Public Shared ReadOnly Property HorizontalScrollBarThumbWidth As Integer

[JScript] public static function get HorizontalScrollBarThumbWidth() : int;

Description

Gets the width, in pixels, of the scroll box in a horizontal scroll bar.

Use

System.Windows.Forms.SystemInformation.HorizontalScrollBarThumbWidth to determine the width of the scroll box used to indicate scroll bar position and to drag the scroll bar to a new position. The scroll box is also called the thumb.

rr) IconSize

ss) ToString

[C#] public static Size IconSize {get;}

[C++] public: __property static Size get_IconSize();

[VB] Public Shared ReadOnly Property IconSize As Size

[JScript] public static function get IconSize() : Size;

Description

Gets the default dimensions, in pixels, of an icon.

Use **System.Windows.Forms.SystemInformation.IconSize** to determine the Windows default icon size. This property determines whether the size of icons displayed in your applications are consistent with icons displayed in Windows.

tt) IconSpacingSize

uu) ToString

[C#] public static Size IconSpacingSize {get;}

[C++] public: __property static Size get_IconSpacingSize();

[VB] Public Shared ReadOnly Property IconSpacingSize As Size

[JScript] public static function get IconSpacingSize() : Size;

Description

Gets the dimensions, in pixels, of the grid used to arrange icons in a large-icon view.

Use **System.Windows.Forms.SystemInformation.IconSpacingSize** to determine the grid that each icon fits into when you arrange them. This value is

always greater than or equal to

System.Windows.Forms.SystemInformation.IconSize .

vv) *KanjiWindowHeight*

ww) *ToString*

[C#] public static int KanjiWindowHeight {get;}

[C++] public: __property static int get_KanjiWindowHeight();

[VB] Public Shared ReadOnly Property KanjiWindowHeight As Integer

[JScript] public static function get KanjiWindowHeight() : int;

Description

Gets the height, in pixels, of the Kanji window at the bottom of the screen for double-byte character set (DBCS) versions of Windows.

Use **System.Windows.Forms.SystemInformation.KanjiWindowHeight** to determine the height of the Kanji window on operating systems that support DBCS. To determine whether the operating system supports DBCS, use the **System.Windows.Forms.SystemInformation.DbcsEnabled** property.

xx) *MaxWindowTrackSize*

yy) *ToString*

[C#] public static Size MaxWindowTrackSize {get;}

[C++] public: __property static Size get_MaxWindowTrackSize();

[VB] Public Shared ReadOnly Property MaxWindowTrackSize As Size

[JScript] public static function get MaxWindowTrackSize() : Size;

Description

Gets the default maximum dimensions, in pixels, of a window that has a caption and sizing borders.

The value returned by

System.Windows.Forms.SystemInformation.MaxWindowTrackSize refers to dimensions of the entire desktop. The user cannot drag the window frame to a size larger than these dimensions. A

System.Windows.Forms.Form can override this value by overriding the **System.Windows.Forms.Form.MaximumSize** property.

zz) *MenuButtonSize*

aaa) *ToString*

[C#] public static Size MenuButtonSize {get;}

[C++] public: __property static Size get_MenuButtonSize();

[VB] Public Shared ReadOnly Property MenuButtonSize As Size

[JScript] public static function get MenuButtonSize() : Size;

Description

Gets the dimensions, in pixels, of menu-bar buttons.

Use this property to determine the size of a menu button, such as the child window close buttons used in a multiple-document-interface application. The dimensions of a menu button can be similar to the dimensions returned for caption buttons using

System.Windows.Forms.SystemInformation.ToolWindowCaptionButtonSize .

bbb) *MenuCheckSize*

ccc) *ToString*

[C#] public static Size MenuCheckSize {get;}

[C++] public: __property static Size get_MenuCheckSize();

[VB] Public Shared ReadOnly Property MenuCheckSize As Size

[JScript] public static function get MenuCheckSize() : Size;

1
2 *Description*

3 Gets the dimensions, in pixels, of the default size of a menu check mark.

4 Use **System.Windows.Forms.SystemInformation.MenuCheckSize** to
5 determine the size of the image used by Windows to display a check mark next
6 to a selected menu item.

6 *ddd) MenuFont*

7 *eee) ToString*

8
9 [C#] public static Font MenuFont {get;}

10 [C++] public: __property static Font* get_MenuFont();

11 [VB] Public Shared ReadOnly Property MenuFont As Font

12 [JScript] public static function get MenuFont() : Font;

13
14 *Description*

15 Gets the operating system font for menus.

16 *fff) MenuHeight*

17 *ggg) ToString*

18
19 [C#] public static int MenuHeight {get;}

20 [C++] public: __property static int get_MenuHeight();

21 [VB] Public Shared ReadOnly Property MenuHeight As Integer

22 [JScript] public static function get MenuHeight() : int;

23
24 *Description*

25 Gets the height of one line of a menu in pixels.

Use **System.Windows.Forms.SystemInformation.MenuHeight** to determine the height that is currently defined by Windows for a menu bar.

hhh) MidEastEnabled

iii) ToString

[C#] public static bool MidEastEnabled {get;}

[C++] public: __property static bool get_MidEastEnabled();

[VB] Public Shared ReadOnly Property MidEastEnabled As Boolean

[JScript] public static function get MidEastEnabled() : Boolean;

Description

Gets a value indicating whether the operating system is enabled for Hebrew and Arabic languages.

jjj) MinimizedWindowSize

kkk) ToString

[C#] public static Size MinimizedWindowSize {get;}

[C++] public: __property static Size get_MinimizedWindowSize();

[VB] Public Shared ReadOnly Property MinimizedWindowSize As Size

[JScript] public static function get MinimizedWindowSize() : Size;

Description

Gets the dimensions, in pixels, of a normal minimized window.

III) *MinimizedWindowSpacingSize*

mmm) ToString

```
[C#]    public    static    Size    MinimizedWindowSpacingSize    {get;}
[C++]   public:  __property static Size get_MinimizedWindowSpacingSize();
[VB]    Public Shared ReadOnly Property MinimizedWindowSpacingSize As Size
[JScript] public static function get MinimizedWindowSpacingSize() : Size;
```

Description

Gets the dimensions, in pixels, of the grid into which minimized windows are placed.

Use

System.Windows.Forms.SystemInformation.MinimizedWindowSpacingSize to determine the rectangle to which a minimized window is sized when arranged. The value of this property is always greater than or equal to the value of the **System.Windows.Forms.SystemInformation.MinimumWindowSize** property.

nnn) MinimumWindowSize

ooo) ToString

```
[C#]    public    static    Size    MinimumWindowSize    {get;}
[C++]   public:  __property static Size get_MinimumWindowSize();
[VB]    Public Shared ReadOnly Property MinimumWindowSize As Size
[JScript] public static function get MinimumWindowSize() : Size;
```

Description

Gets the minimum allowable dimensions, in pixels, of a window.

Use **System.Windows.Forms.SystemInformation.MinimumWindowSize** to determine the dimensions specified by Windows as the minimum size for a window. You can use this property to limit the resizing of the windows of your application to a size larger than the dimension returned by this property.

ppp) MinWindowTrackSize

qqq) ToString

[C#] public static Size MinWindowTrackSize {get;}

[C++] public: __property static Size get_MinWindowTrackSize();

[VB] Public Shared ReadOnly Property MinWindowTrackSize As Size

[JScript] public static function get MinWindowTrackSize() : Size;

Description

Gets the default minimum tracking dimensions, in pixels, of the operating system for a window.

The user cannot drag the window frame to a size smaller than these dimensions. A Form can override these values by overriding the **System.Windows.Forms.Form.MinimumSize** .

rrr) MonitorCount

sss) ToString

[C#] public static int MonitorCount {get;}

[C++] public: __property static int get_MonitorCount();

[VB] Public Shared ReadOnly Property MonitorCount As Integer

[JScript] public static function get MonitorCount() : int;

Description

Gets the number of display monitors on the desktop.

This property returns a value greater than one only if multiple monitors are currently connected to the operating system.

ttt) MonitorsSameDisplayFormat

uuu) ToString

[C#] public static bool MonitorsSameDisplayFormat {get;}

[C++] public: __property static bool get_MonitorsSameDisplayFormat();

[VB] Public Shared ReadOnly Property MonitorsSameDisplayFormat As Boolean

[JScript] public static function get MonitorsSameDisplayFormat() : Boolean;

Description

Gets a value indicating whether all the display monitors have the same color format.

Use

System.Windows.Forms.SystemInformation.MonitorsSameDisplayFormat to determine whether all monitors currently connected to the system have the same color format. For example, the RGB values can be encoded with a different number of bits, or those bits can be located in different places in a pixel's color value.

vvv) MouseButtons

www) ToString

[C#] public static int MouseButtons {get;}

[C++] public: __property static int get_MouseButtons();

[VB] Public Shared ReadOnly Property MouseButtons As Integer

[JScript] public static function get MouseButtons() : int;

Description

Gets the number of buttons on the mouse.

Use **System.Windows.Forms.SystemInformation.MouseButtons** to determine the number of mouse buttons that your application can support. You can also use **System.Windows.Forms.SystemInformation.MouseButtons** to provide additional functionality if more than two buttons are available.

xxx) MouseButtonsSwapped

yyy) ToString

```
[C#]      public      static      bool      MouseButtonsSwapped      {get;}
```

```
[C++]     public:     __property     static     bool     get_MouseButtonsSwapped();
```

```
[VB] Public Shared ReadOnly Property MouseButtonsSwapped As Boolean
```

```
[JScript] public static function get MouseButtonsSwapped() : Boolean;
```

Description

Gets a value indicating whether the functions of the left and right mouse buttons have been swapped.

Use

System.Windows.Forms.SystemInformation.MouseButtonsSwapped to determine whether the left and right mouse buttons have opposite usage. This property determines how the system responds to mouse button clicks and other mouse button events.

zzz) MousePresent

aaaa) ToString

```
[C#]      public      static      bool      MousePresent      {get;}
```

```
[C++]     public:     __property     static     bool     get_MousePresent();
```

```
[VB] Public Shared ReadOnly Property MousePresent As Boolean
```

```
[JScript] public static function get MousePresent() : Boolean;
```

1
2 *Description*

3 Gets a value indicating whether a mouse is installed.

4 *bbbb) MouseWheelPresent*

5 *cccc) ToString*

6
7 [C#] public static bool MouseWheelPresent {get;}

8 [C++] public: __property static bool get_MouseWheelPresent();

9 [VB] Public Shared ReadOnly Property MouseWheelPresent As Boolean

10 [JScript] public static function get MouseWheelPresent() : Boolean;

11
12 *Description*

13 Gets a value indicating whether a mouse with a mouse wheel is installed.

14 Use this property to determine whether to process code that requires a mouse wheel.

15
16 *dddd) MouseWheelScrollLines*

17 *eeee) ToString*

18
19 [C#] public static int MouseWheelScrollLines {get;}

20 [C++] public: __property static int get_MouseWheelScrollLines();

21 [VB] Public Shared ReadOnly Property MouseWheelScrollLines As Integer

22 [JScript] public static function get MouseWheelScrollLines() : int;

23
24 *Description*

25 Gets the number of lines to scroll when the mouse wheel is rotated.

Use this property to determine how many lines to increase or decrease in a control that has a scroll bar.

ffff) NativeMouseWheelSupport

gggg) ToString

[C#] public static bool NativeMouseWheelSupport {get;}

[C++] public: __property static bool get_NativeMouseWheelSupport();

[VB] Public Shared ReadOnly Property NativeMouseWheelSupport As Boolean

[JScript] public static function get NativeMouseWheelSupport() : Boolean;

Description

Gets a value indicating whether the operating system natively supports a mouse wheel.

Mouse wheel operations that occur through a **System.Windows.Forms.Control** object work even if the operating system does not natively support the wheel.

hhhh) Network

iiii) ToString

[C#] public static bool Network {get;}

[C++] public: __property static bool get_Network();

[VB] Public Shared ReadOnly Property Network As Boolean

[JScript] public static function get Network() : Boolean;

Description

Gets a value indicating whether this computer is connected to a network.

1 Use **System.Windows.Forms.SystemInformation.Network** to determine
whether a network is available before performing network-oriented operations.

2 *jjjj) PenWindows*

3 *kkkk) ToString*

4
5 [C#] public static bool PenWindows {get;}

6 [C++] public: __property static bool get_PenWindows();

7 [VB] Public Shared ReadOnly Property PenWindows As Boolean

8 [JScript] public static function get PenWindows() : Boolean;

9
10 *Description*

11 Gets a value indicating whether the Microsoft Windows for Pen Computing
12 extensions are installed.

13 *llll) PrimaryMonitorMaximizedWindowSize*

14 *mmmm)ToString*

15
16 [C#] public static Size PrimaryMonitorMaximizedWindowSize {get;}

17 [C++] public: __property static Size
18 get_PrimaryMonitorMaximizedWindowSize();

19 [VB] Public Shared ReadOnly Property PrimaryMonitorMaximizedWindowSize

20 As Size

21 [JScript] public static function get PrimaryMonitorMaximizedWindowSize() :
22 Size;

23
24 *Description*

Gets the default dimensions, in pixels, of a maximized window on the primary monitor.

nnnn) PrimaryMonitorSize

oooo) ToString

[C#] public static Size PrimaryMonitorSize {get;}

[C++] public: __property static Size get_PrimaryMonitorSize();

[VB] Public Shared ReadOnly Property PrimaryMonitorSize As Size

[JScript] public static function get PrimaryMonitorSize() : Size;

Description

Gets the dimensions, in pixels, of the primary display monitor.

pppp) RightAlignedMenus

qqqq) ToString

[C#] public static bool RightAlignedMenus {get;}

[C++] public: __property static bool get_RightAlignedMenus();

[VB] Public Shared ReadOnly Property RightAlignedMenus As Boolean

[JScript] public static function get RightAlignedMenus() : Boolean;

Description

Gets a value indicating whether drop-down menus are right-aligned with the corresponding menu-bar item.

rrrr) *Secure*

ssss) *ToString*

[C#] public static bool Secure {get;}

[C++] public: __property static bool get_Secure();

[VB] Public Shared ReadOnly Property Secure As Boolean

[JScript] public static function get Secure() : Boolean;

Description

Gets a value indicating whether security is present on this operating system.

System.Windows.Forms.SystemInformation.Secure allows you to determine whether a Security Manager is available from the operating system. Windows NT and Windows 2000 provide a Security Manager to determine access to the operating system registry and to the file system. Windows 95 and 98 do not provide a Security Manager.

tttt) *ShowSounds*

uuuu) *ToString*

[C#] public static bool ShowSounds {get;}

[C++] public: __property static bool get_ShowSounds();

[VB] Public Shared ReadOnly Property ShowSounds As Boolean

[JScript] public static function get ShowSounds() : Boolean;

Description

Gets a value indicating whether the user requires an application to present information in visual form in situations when it would present the information in audible form.

Use **System.Windows.Forms.SystemInformation.ShowSounds** to determine whether the application can play sounds and other audio outputs or whether the audio needs to be visually shown. This property can help provide accessibility to your application.

vvvv) SmallIconSize

www)ToString

[C#] public static Size SmallIconSize {get;}

[C++] public: __property static Size get_SmallIconSize();

[VB] Public Shared ReadOnly Property SmallIconSize As Size

[JScript] public static function get SmallIconSize() : Size;

Description

Gets the recommended dimensions, in pixels, of a small icon.

Use **System.Windows.Forms.SystemInformation.SmallIconSize** to determine the size of small icons in Windows. Small icons typically appear in window captions and in the Small Icon view in Windows Explorer.

xxxx) ToolWindowCaptionButtonSize

yyyy) ToString

[C#] public static Size ToolWindowCaptionButtonSize {get;}

[C++] public: __property static Size get_ToolWindowCaptionButtonSize();

[VB] Public Shared ReadOnly Property ToolWindowCaptionButtonSize As Size

[JScript] public static function get ToolWindowCaptionButtonSize() : Size;

Description

Gets the dimensions, in pixels, of small caption buttons.

Small captions are used in floating tool windows. Use

System.Windows.Forms.SystemInformation.ToolWindowCaptionButton Size to get the dimensions of the buttons placed in the caption of a tool window. You can use this property when adding your own custom buttons to the caption in a tool window.

zzzz) ToolWindowCaptionHeight

aaaaa) ToString

[C#] public static int ToolWindowCaptionHeight {get;}

[C++] public: __property static int get_ToolWindowCaptionHeight();

[VB] Public Shared ReadOnly Property ToolWindowCaptionHeight As Integer

[JScript] public static function get ToolWindowCaptionHeight() : int;

Description

Gets the height, in pixels, of a small caption.

Small captions are used in floating tool windows. Use

System.Windows.Forms.SystemInformation.ToolWindowCaptionHeight to determine the height of a tool window caption in Windows. This property can assist in manipulating the caption area of a tool window.

bbbbbb) UserDomainName

ccccc) ToString

[C#] public static string UserDomainName {get;}

[C++] public: __property static String* get_UserDomainName();

[VB] Public Shared ReadOnly Property UserDomainName As String

[JScript] public static function get UserDomainName() : String;

Description

Gets the name of the user domain.

dddd) UserInteractive

eeee) ToString

[C#] public static bool UserInteractive {get;}

[C++] public: __property static bool get_UserInteractive();

[VB] Public Shared ReadOnly Property UserInteractive As Boolean

[JScript] public static function get UserInteractive() : Boolean;

Description

Gets a value indicating whether the current process is running in user-interactive mode.

This property is **false** only when running as a service process or from inside a Web application. When **System.Windows.Forms.SystemInformation.UserInteractive** is **false**, do not display any modal dialogs or message boxes, as there is no GUI for the user to interact with.

ffff) UserName

ggggg) ToString

[C#] public static string UserName {get;}

[C++] public: __property static String* get_UserName();

[VB] Public Shared ReadOnly Property UserName As String

[JScript] public static function get UserName() : String;

Description

Gets the user name for the current thread (the name of the user currently logged on to the operating system).

hhhhh)VerticalScrollBarArrowHeight

iiii) ToString

```
[C#]    public    static    int    VerticalScrollBarArrowHeight    {get;}
[C++]  public:  __property  static  int  get_VisualScrollBarArrowHeight();
[VB] Public Shared ReadOnly Property VerticalScrollBarArrowHeight As Integer
[Script] public static function get VerticalScrollBarArrowHeight() : int;
```

Description

Gets the height, in pixels, of the arrow bitmap on the vertical scroll bar.

jjjj) VerticalScrollBarThumbHeight

kkkk) ToString

```
[C#]    public    static    int    VerticalScrollBarThumbHeight    {get;}
[C++]  public:  __property  static  int  get_VisualScrollBarThumbHeight();
[VB] Public Shared ReadOnly Property VerticalScrollBarThumbHeight As
Integer
[Script] public static function get VerticalScrollBarThumbHeight() : int;
```

Description

Gets the height, in pixels, of the scroll box in a vertical scroll bar.

Use

System.Windows.Forms.SystemInformation.VerticalScrollBarThumbHeight to determine the height of the box used to indicate scroll bar position and to drag the scroll bar to a new position. The scroll box is also called the thumb.

IIII) *VerticalScrollBarWidth*

mmmmm)ToString

```
[C#]      public      static      int      VerticalScrollBarWidth      {get;}
[C++]     public:     __property static int get_VirtualScrollBarWidth();
[VB]      Public Shared ReadOnly Property VerticalScrollBarWidth As Integer
[JScript] public static function get VerticalScrollBarWidth() : int;
```

Description

Gets the width, in pixels, of the vertical scroll bar.

nnnnn)VirtualScreen

ooooo) ToString

```
[C#]      public      static      Rectangle      VirtualScreen      {get;}
[C++]     public:     __property static Rectangle get_VirtualScreen();
[VB]      Public Shared ReadOnly Property VirtualScreen As Rectangle
[JScript] public static function get VirtualScreen() : Rectangle;
```

Description

Gets the bounds of the virtual screen.

Use **System.Windows.Forms.SystemInformation.VirtualScreen** to determine the bounds of the entire desktop on a multi-monitor system. You can then determine the maximum visual space available on a system that has multiple monitors installed.

1 *ppppp) WorkingArea*

2 *qqqqq) ToString*

3
4 [C#] public static Rectangle WorkingArea {get;}

5 [C++] public: __property static Rectangle get_WorkingArea();

6 [VB] Public Shared ReadOnly Property WorkingArea As Rectangle

7 [JScript] public static function get WorkingArea() : Rectangle;

8
9 *Description*

10 Gets the size, in pixels, of the working area.

11 Use **System.Windows.Forms.SystemInformation.WorkingArea** to
12 determine the bounds of the screen that can be used by applications. The
13 working area is the portion of the screen not hidden by the operating system
14 tray and other top-level windows that are docked to the Windows desktop.

15 TabAlignment enumeration (System.Windows.Forms)

16
17 *a) ToString*

18
19 *Description*

20 Specifies the locations of the tabs in a tab control.

21 This enumeration is used by members such as
22 **System.Windows.Forms.TabControl.Alignment** .

23
24 *b) ToString*

25 [C#] public const TabAlignment Bottom;

[C++] public: const TabAlignment Bottom;

[VB] Public Const Bottom As TabAlignment

[JScript] public var Bottom : TabAlignment;

Description

The tabs are located across the bottom of the control.

c) ToString

[C#] public const TabAlignment Left;

[C++] public: const TabAlignment Left;

[VB] Public Const Left As TabAlignment

[JScript] public var Left : TabAlignment;

Description

The tabs are located along the left edge of the control.

d) ToString

[C#] public const TabAlignment Right;

[C++] public: const TabAlignment Right;

[VB] Public Const Right As TabAlignment

[JScript] public var Right : TabAlignment;

Description

The tabs are located along the right edge of the control.

e) ToString

[C#] public const TabAlignment Top;

[C++]	public:	const	TabAlignment	Top;
[VB]	Public	Const	Top	As TabAlignment
[JScript]	public	var	Top	: TabAlignment;

Description

The tabs are located across the top of the control.

TabAppearance enumeration (System.Windows.Forms)

a) ToString

Description

Specifies the appearance of the tabs in a tab control.

This enumeration is used by members such as
System.Windows.Forms.TabControl.Appearance .

b) ToString

[C#]	public	const	TabAppearance	Buttons;
[C++]	public:	const	TabAppearance	Buttons;
[VB]	Public	Const	Buttons	As TabAppearance
[JScript]	public	var	Buttons	: TabAppearance;

Description

The tabs have the appearance of three-dimensional buttons.

c) ToString

[C#]	public	const	TabAppearance	FlatButtons;
[C++]	public:	const	TabAppearance	FlatButtons;
[VB]	Public	Const	FlatButtons	As TabAppearance
[JScript]	public	var	FlatButtons	: TabAppearance;

Description

The tabs have the appearance of flat buttons.

d) ToString

[C#]	public	const	TabAppearance	Normal;
[C++]	public:	const	TabAppearance	Normal;
[VB]	Public	Const	Normal	As TabAppearance
[JScript]	public	var	Normal	: TabAppearance;

Description

The tabs have the standard appearance of tabs.

TabControl class (System.Windows.Forms)

a) ToString

Description

Manages a related set of tab pages.

1 A **System.Windows.Forms.TabControl** contains tab pages, which are
2 represented by **System.Windows.Forms.TabPage** objects that you add
3 through the **System.Windows.Forms.Control.Controls** property.

4 *b) TabControl*

5 *Example Syntax:*

6 *c) ToString*

7 [C#] public TabControl();

8 [C++] public: TabControl();

9 [VB] Public Sub New()

10 [JScript] public function TabControl();

11
12 *Description*

13 Initializes a new instance of the **System.Windows.Forms.TabControl** class.

14 *d) AccessibilityObject*

15 *e) AccessibleDefaultActionDescription*

16 *f) AccessibleDescription*

17 *g) AccessibleName*

18 *h) AccessibleRole*

19 *i) Alignment*

20 *j) ToString*

21
22
23 *Description*

24 Gets or sets the area of the control (for example, along the top) where the tabs
25 are aligned.

When the **System.Windows.Forms.TabControl.Alignment** property is set to **Left** or **Right** , the **System.Windows.Forms.TabControl.Multiline** property is automatically set to **true** .

k) *AllowDrop*

l) *Anchor*

m) *Appearance*

n) *ToString*

Description

Gets or sets the visual appearance of the control's tabs.

When you set the **System.Windows.Forms.TabControl.Appearance** property to **FlatButtons** , it only appears as such when the **System.Windows.Forms.TabControl.Alignment** property is set to **Top** . Otherwise, the **System.Windows.Forms.TabControl.Appearance** property displays as if set to the **Buttons** value.

o) *BackColor*

p) *ToString*

```
[C#]      public      override      Color      BackColor      {get;      set;}
```

```
[C++] public: __property virtual Color get_BackColor();public: __property virtual  
void set_BackColor(Color);
```

```
[VB]      Overrides      Public      Property      BackColor      As      Color
```

```
[JScript] public function get BackColor() : Color;public function set  
BackColor(Color);
```

Description

q) *BackgroundImage*

r) *ToString*

[C#] public override Image BackgroundImage {get; set;}

[C++] public: __property virtual Image* get_BackgroundImage();public:

__property virtual void set_BackgroundImage(Image*);

[VB] Overrides Public Property BackgroundImage As Image

[JScript] public function get BackgroundImage() : Image;public function set

BackgroundImage(Image);

Description

- s) *BindingContext*
- t) *Bottom*
- u) *Bounds*
- v) *CanFocus*
- w) *CanSelect*
- x) *Capture*
- y) *CausesValidation*
- z) *ClientRectangle*
- aa) *ClientSize*
- bb) *CompanyName*
- cc) *Container*
- dd) *ContainsFocus*
- ee) *ContextMenu*
- ff) *Controls*
- gg) *Created*
- hh) *CreateParams*
- ii) *ToString*

Description

Returns the parameters needed to create the handle. Inheriting classes can override this to provide extra functionality. They should not, however, forget to call `base.CreateParams()` first to get the struct filled up with the basic info.

1 *jj) Cursor*

2 *kk) DataBindings*

3 *ll) DefaultImeMode*

4 *mm) DefaultSize*

5 *nn) ToString*

6
7
8 *Description*

9 Deriving classes can override this to configure a default size for their control.
10 This is more efficient than setting the size in the control's constructor.

11 *oo) DesignMode*

12 *pp) DisplayRectangle*

13 *qq) ToString*

14
15
16 *Description*

17 Gets the display area of the control's tab pages.

18 *rr) Disposing*

19 *ss) Dock*

20 *tt) DrawMode*

21 *uu) ToString*

22
23
24 *Description*

25 Gets or sets the way that the control's tab pages are drawn.

1 Tab pages can be drawn by the control or by the user of the control. The
2 **System.Windows.Forms.TabControl.DrawMode** property allows users of
the control to customize the way that the tab control is drawn.

3 vv) *Enabled*

4 ww) *Events*

5 xx) *Focused*

6 yy) *Font*

7 zz) *FontHeight*

8 aaa) *ForeColor*

9 bbb) *ToString*

10
11
12 *Description*

13
14 ccc) *Handle*

15 ddd) *HasChildren*

16 eee) *Height*

17 fff) *HotTrack*

18 ggg) *ToString*

19
20
21
22 *Description*

23 Gets or sets a value indicating whether the control's tabs change in appearance
24 when the mouse passes over them.
25

hhh) *ImageList*

iii) *ToString*

```
[C#]      public      ImageList      ImageList      {get;      set;}
[C++] public: __property ImageList* get_ImageList();public: __property void
set_ImageList(ImageList*);
[VB]      Public      Property      ImageList      As      ImageList
[JScript] public function get ImageList() : ImageList;public function set
ImageList(ImageList);
```

Description

Gets or sets the images to display on the control's tabs.

To display an image on a tab, set the **System.Windows.Forms.TabPage.ImageIndex** property of that **System.Windows.Forms.TabPage** . The **System.Windows.Forms.TabPage.ImageIndex** acts as the index into the **System.Windows.Forms.TabControl.ImageList** .

1 *jjj) ImeMode*

2 *kkk) InvokeRequired*

3 *lll) IsAccessible*

4 *mmm) IsDisposed*

5 *nnn) IsHandleCreated*

6 *ooo) ItemSize*

7 *ppp) ToString*

10 *Description*

11 Gets or sets the size of the control's tabs.

12 To change the **System.Drawing.Size.Width** property of the
13 **System.Windows.Forms.TabControl.ItemSize** property, the
14 **System.Windows.Forms.TabControl.SizeMode** property must be set to
15 **Fixed** .

15 *qqq) Left*

16 *rrr) Location*

17 *sss) Multiline*

18 *ttt) ToString*

21 *Description*

22 Gets or sets a value indicating whether more than one row of tabs can be
23 displayed.

24 If **System.Windows.Forms.TabControl.Multiline** is **false** , only one row of
25 tabs is displayed - even if all the tabs do not fit in the available space. In that
 case, however, arrows are displayed that allow the user to navigate to the
 undisplayed tabs.

uuu) *Name*

vvv) *Padding*

www) *ToString*

Description

Gets or sets the amount of space around each item on the control's tab pages.

xxx) *Parent*

yyy) *ProductName*

zzz) *ProductVersion*

aaaa) *RecreatingHandle*

bbbb) *Region*

cccc) *RenderRightToLeft*

dddd) *ResizeRedraw*

eeee) *Right*

ffff) *RightToLeft*

gggg) *RowCount*

hhhh) *ToString*

Description

Gets the number of rows that are currently being displayed in the control's tab strip.

Use the **System.Windows.Forms.TabControl.RowCount** property when the **System.Windows.Forms.TabControl.Multiline** property is **true** and you want to know the number of rows that the tabs occupy.

iiii) SelectedIndex

jjjj) ToString

[C#] public int SelectedIndex {get; set;}

[C++] public: __property int get_SelectedIndex();public: __property void
set_SelectedIndex(int);

[VB] Public Property SelectedIndex As Integer

[JScript] public function get SelectedIndex() : int;public function set
SelectedIndex(int);

Description

Gets or sets the index of the currently-selected tab page.

kkkk) SelectedTab

llll) ToString

[C#] public TabPage SelectedTab {get; set;}

[C++] public: __property TabPage* get_SelectedTab();public: __property void
set_SelectedTab(TabPage*);

[VB] Public Property SelectedTab As TabPage

[JScript] public function get SelectedTab() : TabPage;public function set
SelectedTab(TabPage);

1
2 *Description*

3 Gets or sets the currently-selected tab page.

4 The selection to the given tab, provided it equals a tab in the list.

5 *mmmm)ShowFocusCues*

6 *nnnn) ShowKeyboardCues*

7 *oooo) ShowToolTips*

8 *pppp) ToString*

9
10
11 *Description*

12 Gets or sets a value indicating whether a tab's ToolTip is shown when the mouse
13 passes over the tab.

14 To create a ToolTip for a tab, set the
15 **System.Windows.Forms.TabPage.ToolTipText** property of the
16 **System.Windows.Forms.TabPage** .

17 *qqqq) Site*

18 *rrrr) Size*

19 *ssss) SizeMode*

20 *tttt) ToString*

21
22 *Description*

23 Gets or sets the way that the control's tabs are sized.

uuuu) *TabCount*

vvvv) *ToString*

[C#]	public	int	TabCount	{get;}
[C++]	public:	__property	int	get_TabCount();
[VB]	Public	ReadOnly	Property	TabCount As Integer
[JScript]	public	function	get	TabCount() : int;

Description

Gets the number of tabs in the tab strip.

www) *TabIndex*

xxxx) *TabPages*

yyyy) *ToString*

Description

Gets the collection of tab pages in this tab control.

zzzz) *TabStop*

aaaaa) *Tag*

bbbbbb) *Text*

cccc) *ToString*

dddd) *Top*

eeee) *TopLevelControl*

ffff) *Visible*

ggggg) *Width*

hhhhh) *WindowTarget*

iiii) *ToString*

Description

jjjj) *ToString*

Description

Occurs when the tabs are drawn, if the **System.Windows.Forms.TabControl.DrawMode** property is set to **OwnerDrawFixed** .

For more information about handling events, see .

kkkkk) ToString

Description

Occurs when the **System.Windows.Forms.TabControl.SelectedIndex** property is changed.

For more information about handling events, see .

lllll) CreateControlsInstance

[C#] protected override ControlCollection CreateControlsInstance();

[C++] protected: ControlCollection* CreateControlsInstance();

[VB] Overrides Protected Function CreateControlsInstance() As ControlCollection

[JScript] protected override function CreateControlsInstance() : ControlCollection;

Description

mmmmm) CreateHandle

[C#] protected override void CreateHandle();

[C++] protected: void CreateHandle();

[VB] Overrides Protected Sub CreateHandle()

[JScript] protected override function CreateHandle();

Description

nnnnn)GetControl

```
1
2 [C#]      public      Control      GetControl(int      index);
3
4 [C++]      public:      Control*      GetControl(int      index);
5
6 [VB] Public Function GetControl(ByVal index As Integer) As Control
7
8 [JScript] public function GetControl(index : int) : Control;
```

Description

ooooo) GetItems

```
9
10
11 [C#]      protected      virtual      object[]      GetItems();
12
13 [C++]      protected:      virtual      Object*      GetItems()      __gc[];
14
15 [VB] Overridable Protected Function GetItems() As Object()
16
17 [JScript] protected function GetItems() : Object[];
```

Description

This has package scope so that TabStrip and TabControl can call it.

ppppp) GetItems

```
18
19
20 [C#]      protected      virtual      object[]      GetItems(Type      baseType);
21
22 [C++]      protected:      virtual      Object*      GetItems(Type*      baseType)      __gc[];
23
24 [VB] Overridable Protected Function GetItems(ByVal baseType As Type) As
25 Object()
26
27 [JScript] protected function GetItems(baseType : Type) : Object[];
```

Description

This has package scope so that TabStrip and TabControl can call it.

qqqqq) GetTabRect

```
[C#]      public      Rectangle      GetTabRect(int      index);
[C++]      public:      Rectangle      GetTabRect(int      index);
[VB] Public Function GetTabRect(ByVal index As Integer) As Rectangle
[JScript] public function GetTabRect(index : int) : Rectangle;
```

Description

Returns the bounding rectangle for a specified tab in this tab control.

Return Value: A **System.Drawing.Rectangle** that represents the bounds of the specified tab. The 0-based index of the tab you want.

rrrrr) GetToolTipText

```
[C#]      protected      string      GetToolTipText(object      item);
[C++]      protected:      String*      GetToolTipText(Object*      item);
[VB] Protected Function GetToolTipText(ByVal item As Object) As String
[JScript] protected function GetToolTipText(item : Object) : String;
```

Description

sssss) IsInputKey

```
[C#]      protected      override      bool      IsInputKey(Keys      keyData);
```

1 [C++] protected: bool IsInputKey(Keys keyData);

2 [VB] Overrides Protected Function IsInputKey(ByVal keyData As Keys) As
3 Boolean

4 [JScript] protected override function IsInputKey(keyData : Keys) : Boolean;

6 *Description*

7 Determines whether the specified key is a regular input key or a special key that
8 requires preprocessing.

9 *Return Value:* **true** if the specified key is a regular input key; otherwise, **false** .

10 Call this method during window-message preprocessing to determine whether
11 the specified key is a regular input key that should be sent directly to the tab
12 control or a special key (such as PAGE UP and PAGE DOWN) that should
13 be preprocessed. In the latter case, send the key to the control only if it is not
14 consumed by the preprocessing phase. One of the
15 **System.Windows.Forms.Keys** values.

13 *ttttt) OnDrawItem*

15 [C#] protected virtual void OnDrawItem(DrawItemEventArgs e);

16 [C++] protected: virtual void OnDrawItem(DrawItemEventArgs* e);

17 [VB] Overridable Protected Sub OnDrawItem(ByVal e As DrawItemEventArgs)

18 [JScript] protected function OnDrawItem(e : DrawItemEventArgs);

20 *Description*

21 Raises the **System.Windows.Forms.TabControl.DrawItem** event.

22 Raising an event invokes the event handler through a delegate. For more
23 information, see . A **System.Windows.Forms.DrawItemEventArgs** that
24 contains the event data.

uuuuu)OnFontChanged

```
[C#]    protected    override    void    OnFontChanged(EventArgs    e);
[C++]    protected:    void    OnFontChanged(EventArgs*    e);
[VB]    Overrides Protected Sub OnFontChanged(ByVal e As EventArgs)
[JScript] protected override function OnFontChanged(e : EventArgs);
```

Description

Raises the **System.Windows.Forms.Control.FontChanged** event.

Raising an event invokes the event handler through a delegate. For more information, see . An **System.EventArgs** that contains the event data.

vvvvv) OnHandleCreated

```
[C#]    protected    override    void    OnHandleCreated(EventArgs    e);
[C++]    protected:    void    OnHandleCreated(EventArgs*    e);
[VB]    Overrides Protected Sub OnHandleCreated(ByVal e As EventArgs)
[JScript] protected override function OnHandleCreated(e : EventArgs);
```

Description

This is a notification that the handle has been created. We do some work here to configure the handle. Overriders should call base.OnHandleCreated() This is a notification that the handle has been created. We do some work here to configure the handle. Overriders should call base.OnHandleCreated()

wwwww)OnHandleDestroyed

```
[C#]    protected    override    void    OnHandleDestroyed(EventArgs    e);
[C++]    protected:    void    OnHandleDestroyed(EventArgs*    e);
```

1 [VB] Overrides Protected Sub OnHandleDestroyed(ByVal e As EventArgs)

2 [JScript] protected override function OnHandleDestroyed(e : EventArgs);

3
4 *Description*

5 Raises the **System.Windows.Forms.Control.HandleDestroyed** event.

6 Raising an event invokes the event handler through a delegate. For more
7 information, see . An **System.EventArgs** that contains the event data.

8
9 *xxxxx) OnKeyDown*

10 [C#] protected override void OnKeyDown(KeyEventArgs ke);

11 [C++] protected: void OnKeyDown(KeyEventArgs* ke);

12 [VB] Overrides Protected Sub OnKeyDown(ByVal ke As KeyEventArgs)

13 [JScript] protected override function OnKeyDown(ke : KeyEventArgs);

14
15 *Description*

16 We override this to get tabbing functionality. If overriding this, remember to call
17 base.onKeyDown.

18
19 *yyyyy) OnResize*

20 [C#] protected override void OnResize(EventArgs e);

21 [C++] protected: void OnResize(EventArgs* e);

22 [VB] Overrides Protected Sub OnResize(ByVal e As EventArgs)

23 [JScript] protected override function OnResize(e : EventArgs);

24
25 *Description*

zzzzz) OnSelectedIndexChanged

```
[C#]    protected    virtual    void    OnSelectedIndexChanged(EventArgs e);
[C++]    protected:    virtual    void    OnSelectedIndexChanged(EventArgs* e);
[VB]    Overridable    Protected    Sub    OnSelectedIndexChanged(ByVal e As
EventArgs)
[JScript]    protected    function    OnSelectedIndexChanged(e : EventArgs);
```

Description

Raises the **System.Windows.Forms.TabControl.SelectedIndexChanged** event.

Raising an event invokes the event handler through a delegate. For more information, see . An **System.EventArgs** that contains the event data.

aaaaaa) OnStyleChanged

```
[C#]    protected    override    void    OnStyleChanged(EventArgs e);
[C++]    protected:    void    OnStyleChanged(EventArgs* e);
[VB]    Overrides    Protected    Sub    OnStyleChanged(ByVal e As EventArgs)
[JScript]    protected    override    function    OnStyleChanged(e : EventArgs);
```

Description

bbbbbb) ProcessKeyPreview

```
[C#]    protected    override    bool    ProcessKeyPreview(ref Message m);
[C++]    protected:    bool    ProcessKeyPreview(Message* m);
[VB]    Overrides    Protected    Function    ProcessKeyPreview(ByRef m As Message) As
```

Boolean

[JScript] protected override function ProcessKeyPreview(m : Message) : Boolean;

Description

We override this to get the Ctrl and Ctrl-Shift Tab functionality.

cccccc)RemoveAll

[C#]	protected	void	RemoveAll();
------	-----------	------	--------------

[C++]	protected:	void	RemoveAll();
-------	------------	------	--------------

[VB]	Protected	Sub	RemoveAll()
------	-----------	-----	-------------

[JScript]	protected	function	RemoveAll();
-----------	-----------	----------	--------------

Description

Removes all the tab pages and additional controls from this tab control.

All controls are removed through the Controls property.

dddddd)ToString

[C#]	public	override	string	ToString();
------	--------	----------	--------	-------------

[C++]	public:	String*	ToString();
-------	---------	---------	-------------

[VB]	Overrides	Public	Function	ToString()	As	String
------	-----------	--------	----------	------------	----	--------

[JScript]	public	override	function	ToString()	:	String;
-----------	--------	----------	----------	------------	---	---------

Description

Returns a string representation for this control.

Return Value: String Returns a string representation for this control.

eeeeee)UpdateTabSelection

[C#] protected void UpdateTabSelection(bool uiselectd);
[C++] protected: void UpdateTabSelection(bool uiselectd);
[VB] Protected Sub UpdateTabSelection(ByVal uiselectd As Boolean)
[JScript] protected function UpdateTabSelection(uiselectd : Boolean);

Description

Description

Set the panel selections appropriately Set the panel selections appropriately

ffffff) WndProc

[C#] protected override void WndProc(ref Message m);
[C++] protected: void WndProc(Message* m);
[VB] Overrides Protected Sub WndProc(ByRef m As Message)
[JScript] protected override function WndProc(m : Message);

Description

The tab's window procedure. Inheritng classes can override this to add extra functionality, but should not forget to call base.wndProc(m); to ensure the tab continues to function properly. A Windows Message Object.

TabDrawMode enumeration (System.Windows.Forms)

a) *WndProc*

Description

Specifies whether the tabs in a tab control are owner-drawn (drawn by the parent window), or drawn by the operating system.

This enumeration is used by members such as
System.Windows.Forms.TabControl.DrawMode .

b) *WndProc*

[C#]	public	const	TabDrawMode	Normal;
[C++]	public:	const	TabDrawMode	Normal;
[VB]	Public	Const	Normal	As TabDrawMode
[JScript]	public	var	Normal	: TabDrawMode;

Description

The tabs are drawn by the operating system, and are of the same size.

c) *WndProc*

[C#]	public	const	TabDrawMode	OwnerDrawFixed;
[C++]	public:	const	TabDrawMode	OwnerDrawFixed;
[VB]	Public	Const	OwnerDrawFixed	As TabDrawMode
[JScript]	public	var	OwnerDrawFixed	: TabDrawMode;

1
2 *Description*

3 The tabs are drawn by the parent window, and are of the same size.

4 TabPage class (System.Windows.Forms)

5 a) *ToString*

6
7
8 *Description*

9 Represents a single tab page in a **System.Windows.Forms.TabControl** .

10 For more information about how this control responds to the
11 **System.Windows.Forms.Control.Focus** and
12 **System.Windows.Forms.Control.Select** methods, see the following
13 **System.Windows.Forms.Control** members:
14 **System.Windows.Forms.Control.CanFocus** ,
15 **System.Windows.Forms.Control.CanSelect** ,
16 **System.Windows.Forms.Control.Focused** ,
17 **System.Windows.Forms.Control.ContainsFocus** ,
18 **System.Windows.Forms.Control.Focus** ,
19 **System.Windows.Forms.Control.Select** .

20 b) *TabPage*

21 *Example Syntax:*

22 c) *ToString*

23 [C#] public TabPage();
24 [C++] public: TabPage();
25 [VB] Public Sub New()
26 [JScript] public function TabPage(); Initializes a new instance of the
27 **System.Windows.Forms.TabPage** class.

Description

Initializes a new instance of the **System.Windows.Forms.TabPage** class.

d) TabPage

Example Syntax:

e) ToString

[C#] public TabPage(string text);

[C++] public: TabPage(String* text);

[VB] Public Sub New(ByVal text As String)

[JScript] public function TabPage(text : String);

Description

Initializes a new instance of the **System.Windows.Forms.TabPage** class, specifies the text for the tab.

The **System.Windows.Forms.TabPage.Text** property is set to the value of the *text* parameter. The text for the tab.

- f) *AccessibilityObject*
- g) *AccessibleDefaultActionDescription*
- h) *AccessibleDescription*
- i) *AccessibleName*
- j) *AccessibleRole*
- k) *AllowDrop*
- l) *Anchor*
- m) *ToString*

Description

Gets or sets the way that the tab page anchors to the edges of its container.

When a tab page is anchored to an edge of a tab control, the distance between the tab page and the specified edge remains constant when the tab control resizes. For example, if a tab page is anchored to the right edge of its tab control, the distance between the right edge of the tab page and the right edge of the tab control remains constant when the tab control resizes. A tab page can be anchored to any combination of edges. If the tab page is anchored to opposite edges of its container (for example, the top and bottom), it resizes when the tab control resizes. If a tab page has its

System.Windows.Forms.Control.Anchor property set to **AnchorStyle.None**, the tab page moves half of the distance that the tab control is resized. For example, if a **System.Windows.Forms.TabPage** has its **System.Windows.Forms.TabPage.Anchor** property set to **AnchorStyle.None** and the **System.Windows.Forms.TabControl** that the tab page is located on is resized by 20 pixels in either direction, the button will be moved 10 pixels in both directions.

- 1 ***n) AutoScroll***
- 2 ***o) AutoScrollMargin***
- 3 ***p) AutoScrollMinSize***
- 4 ***q) AutoScrollPosition***
- 5 ***r) BackColor***
- 6 ***s) BackgroundImage***
- 7 ***t) BindingContext***
- 8 ***u) BorderStyle***
- 9 ***v) Bottom***
- 10 ***w) Bounds***
- 11 ***x) CanFocus***
- 12 ***y) CanSelect***
- 13 ***z) Capture***
- 14 ***aa) CausesValidation***
- 15 ***bb) ClientRectangle***
- 16 ***cc) ClientSize***
- 17 ***dd) CompanyName***
- 18 ***ee) Container***
- 19 ***ff) ContainsFocus***
- 20 ***gg) ContextMenu***
- 21 ***hh) Controls***
- 22 ***ii) Created***
- 23 ***jj) CreateParams***
- 24
- 25

1 **kk) *Cursor***

2 **ll) *DataBindings***

3 **mm) *DefaultImeMode***

4 **nn) *DefaultSize***

5 **oo) *DesignMode***

6 **pp) *DisplayRectangle***

7 **qq) *Disposing***

8 **rr) *Dock***

9 **ss) *ToString***

10
11
12 ***Description***

13 Gets or sets the position and manner in which the tab page is docked.

14 When a control is docked to an edge of it's container, it will always be positioned
15 flush against that edge when the container is resized. If more than one control is
16 docked to an edge, the controls will not be placed on top of each other.

17 **tt) *DockPadding***

18 **uu) *Enabled***

19 **vv) *ToString***

20
21
22 ***Description***

23 Gets or sets a value indicating whether the tab is enabled.

1 *ww) Events*
2 *xx) Focused*
3 *yy) Font*
4 *zz) FontHeight*
5 *aaa) ForeColor*
6 *bbb) Handle*
7 *ccc) HasChildren*
8 *ddd) Height*
9 *eee) HScroll*
10 *fff) ImageIndex*
11 *ggg) ToString*
12
13
14

15 *Description*

16 Gets or sets the index to the image displayed on this tab.

17 The **System.Windows.Forms.TabPage.ImageIndex** points to an image in
18 the **System.Windows.Forms.TabControl** object's associated
19 **System.Windows.Forms.TabControl.ImageList** .
20
21
22
23
24
25

1 **hhh) ImeMode**
2 **iii) InvokeRequired**
3 **jjj) IsAccessible**
4 **kkk) IsDisposed**
5 **lll) IsHandleCreated**
6 **mmm) Left**
7 **nnn) Location**
8 **ooo) Name**
9 **ppp) Parent**
10 **qqq) ProductName**
11 **rrr) ProductVersion**
12 **sss) RecreatingHandle**
13 **ttt) Region**
14 **uuu) RenderRightToLeft**
15 **vvv) ResizeRedraw**
16 **www) Right**
17 **xxx) RightToLeft**
18 **yyy) ShowFocusCues**
19 **zzz) ShowKeyboardCues**
20 **aaaa) Site**
21 **bbbb) Size**
22 **cccc) TabIndex**
23 **dddd) ToString**

1
2
3 *Description*

4 The index of this tab page into the **System.Windows.Forms.TabControl**
object's tab-page collection.

5 You can access the tab-page collection through the
6 **System.Windows.Forms.TabControl.TabPages** property of the
7 **System.Windows.Forms.TabControl** .

8 *eeee) TabStop*

9 *ffff) ToString*

10
11 [C#] public new bool TabStop {get; set;}

12 [C++] public: __property bool get_TabStop();public: __property void
13 set_TabStop(bool);

14 [VB] Public Property TabStop As Boolean

15 [JScript] public function get TabStop() : Boolean;public function set
16 TabStop(Boolean);

17
18 *Description*

19 Gets or sets a value indicating whether the focus moves to this tab page when
20 the user presses the TAB key.
21
22
23
24
25

gggg) Tag

hhhh) Text

iiii) ToString

Description

Gets or sets the text to display on the tab.

jjjj) ToolTipText

kkkk) ToString

[C#] public string ToolTipText {get; set;}

[C++] public: __property String* get_ToolTipText();public: __property void
set_ToolTipText(String*);

[VB] Public Property ToolTipText As String

[JScript] public function get ToolTipText() : String;public function set
ToolTipText(String);

Description

Gets or sets the ToolTip text for this tab.

This tab page belongs to a **System.Windows.Forms.TabControl** instance.
The ToolTip text appears when the user moves the mouse over the tab - if the
System.Windows.Forms.TabControl.ShowToolTips property of the
System.Windows.Forms.TabControl is **true** . For more information on
ToolTips, see the **System.Windows.Forms.ToolTip** class.

1 *llll) Top*

2 *mmmm)TopLevelControl*

3 *nnnn) Visible*

4 *oooo) ToString*

5
6
7 *Description*

8 Gets or sets a value indicating whether the tab is visible.

9 *pppp) VScroll*

10 *qqqq) Width*

11 *rrrr) WindowTarget*

12 *ssss) CreateControlsInstance*

13
14 [C#] protected override ControlCollection CreateControlsInstance();

15 [C++] protected: ControlCollection* CreateControlsInstance();

16 [VB] Overrides Protected Function CreateControlsInstance() As ControlCollection

17 [JScript] protected override function CreateControlsInstance() : ControlCollection;

18
19 *Description*

20 Constructs the new instance of the Controls collection objects. Subclasses should
21 not call base.CreateControlsInstance. Our version creates a control collection that
22 does not support Constructs the new instance of the Controls collection objects.
23 Subclasses should not call base.CreateControlsInstance. Our version creates a
24 control collection that does not support
25

tttt) *GetTabPageOfComponent*

```
[C#] public static TabPage GetTabPageOfComponent(object comp);  
[C++] public: static TabPage* GetTabPageOfComponent(Object* comp);  
[VB] Public Shared Function GetTabPageOfComponent(ByVal comp As Object)  
As TabPage  
[JScript] public static function GetTabPageOfComponent(comp : Object) :  
TabPage;
```

Description

Retrieves the tab page that contains the specified object.

Return Value: The **System.Windows.Forms.TabPage** that contains the specified object, or **null** if the object cannot be found. The object to look for.

uuuu) *SetBoundsCore*

```
[C#] protected override void SetBoundsCore(int x, int y, int width, int height,  
BoundsSpecified specified);  
[C++] protected: void SetBoundsCore(int x, int y, int width, int height,  
BoundsSpecified specified);  
[VB] Overrides Protected Sub SetBoundsCore(ByVal x As Integer, ByVal y As  
Integer, ByVal width As Integer, ByVal height As Integer, ByVal specified As  
BoundsSpecified)  
[JScript] protected override function SetBoundsCore(x : int, y : int, width : int,  
height : int, specified : BoundsSpecified);
```

Description

overrides main setting of our bounds so that we can control our size and that of our TabPages...

vvvv) ToString

[C#] public override string ToString();

[C++] public: String* ToString();

[VB] Overrides Public Function ToString() As String

[JScript] public override function ToString() : String;

Description

Returns a string containing the value of the **System.Windows.Forms.TabPage.Text** property for the **System.Windows.Forms.TabPage** object's default printing.

Return Value: A string containing the value of the **System.Windows.Forms.TabPage.Text** property.

TabControl.TabPageCollection class (System.Windows.Forms)

a) WndProc

Description

Contains a collection of **System.Windows.Forms.TabPage** objects.

b) TabControl.TabPageCollection

Example Syntax:

c) WndProc

[C#] public TabControl.TabPageCollection(TabControl owner);

[C++] public: TabPageCollection(TabControl* owner);

1 [VB] Public Sub New(ByVal owner As TabControl)

2 [JScript] public function TabControl.TabPageCollection(owner : TabControl);

3
4 *Description*

5 Initializes a new instance of the
6 **System.Windows.Forms.TabControl.TabPageCollection** class. The
7 **System.Windows.Forms.TabControl** that this collection belongs to.

8 d) *Count*

9 e) *WndProc*

10 [C#] public int Count {get;}

11 [C++] public: __property int get_Count();

12 [VB] Public ReadOnly Property Count As Integer

13 [JScript] public function get Count() : int;

14
15 *Description*

16 Gets the number of tab pages in the collection.

17 f) *IsReadOnly*

18 g) *WndProc*

19
20 [C#] public bool IsReadOnly {get;}

21 [C++] public: __property bool get_IsReadOnly();

22 [VB] Public ReadOnly Property IsReadOnly As Boolean

23 [JScript] public function get IsReadOnly() : Boolean;

Description

Gets a value indicating whether the collection is read-only.

h) Item

i) WndProc

```
[C#] public virtual TabPage this[int index] {get; set;}
```

```
[C++] public: __property virtual TabPage* get_Item(int index);public: __property
```

```
virtual void set_Item(int index, TabPage*);
```

```
[VB] Overridable Public Default Property Item(ByVal index As Integer) As
```

```
TabPage
```

```
[JScript] returnValue =
```

```
TabPageCollectionObject.Item(index);TabPageCollectionObject.Item(index) =
```

```
returnValue;
```

Description

Gets or sets a **System.Windows.Forms.TabPage** in the collection. The 0-based index of the tab page to get or set.

j) Add

```
[C#] public void Add(TabPage value);
```

```
[C++] public: void Add(TabPage* value);
```

```
[VB] Public Sub Add(ByVal value As TabPage)
```

```
[JScript] public function Add(value : TabPage);
```

1
2 *Description*

3 Adds a **System.Windows.Forms.TabPage** to the collection. The
4 **System.Windows.Forms.TabPage** to add.

5 *k) AddRange*

6 [C#] public void AddRange(TabPage[] pages);
7 [C++] public: void AddRange(TabPage* pages[]);
8 [VB] Public Sub AddRange(ByVal pages() As TabPage)
9 [JScript] public function AddRange(pages : TabPage[]);
10

11 *Description*

12 Adds a set of tab pages to the collection. An array of type
13 **System.Windows.Forms.TabPage** that contains the tab pages to add.

14 *l) Clear*

15
16 [C#] public virtual void Clear();
17 [C++] public: virtual void Clear();
18 [VB] Overridable Public Sub Clear()
19 [JScript] public function Clear();
20

21 *Description*

22 Removes all the tab pages from the collection.
23
24
25

m) Contains

```
[C#]          public          bool          Contains(TabPage          page);
[C++]          public:          bool          Contains(TabPage*          page);
[VB] Public Function Contains(ByVal page As TabPage) As Boolean
[JScript] public function Contains(page : TabPage) : Boolean;
```

Description

Determines whether a specified tab page is in the collection.

Return Value: **true** if the specified **System.Windows.Forms.TabPage** is in the collection; otherwise, **false** . The **System.Windows.Forms.TabPage** to locate in the collection.

n) GetEnumerator

```
[C#]          public          IEnumerator          GetEnumerator();
[C++]          public:          __sealed          IEnumerator*          GetEnumerator();
[VB] NotOverridable Public Function GetEnumerator() As IEnumerator
[JScript] public function GetEnumerator() : IEnumerator;
```

Description

Returns an enumeration of all the tab pages in the collection.

Return Value: An **System.Collections.IEnumerator** for the **System.Windows.Forms.TabControl.TabPageCollection** .

This method creates an enumerator that contains a snapshot of the collection. You can change the collection by changing the enumerator; however, multiple enumerators can simultaneously access the same collection. Changing the collection (either directly or through another enumerator) can thus cause **System.Collections.IEnumerator.Current** or **System.Collections.IEnumerator.MoveNext** to throw an exception.

o) *IndexOf*

```
[C#]      public      int      IndexOf(TabPage      page);
[C++]      public:      int      IndexOf(TabPage*      page);
[VB] Public Function IndexOf(ByVal page As TabPage) As Integer
[JScript] public function IndexOf(page : TabPage) : int;
```

Description

Returns the index of the specified tab page in the collection.

Return Value: The 0-based index of the tab page; -1 if it cannot be found. The **System.Windows.Forms.TabPage** to locate in the collection.

p) *Remove*

```
[C#]      public      void      Remove(TabPage      value);
[C++]      public:      void      Remove(TabPage*      value);
[VB] Public Sub Remove(ByVal value As TabPage)
[JScript] public function Remove(value : TabPage);
```

Description

Removes a **System.Windows.Forms.TabPage** from the collection. The **System.Windows.Forms.TabPage** to remove.

q) *RemoveAt*

```
[C#]      public      void      RemoveAt(int      index);
[C++]      public:      __sealed void      RemoveAt(int      index);
[VB] NotOverridable Public Sub RemoveAt(ByVal index As Integer)
```

1 [JScript] public function RemoveAt(index : int);

3 *Description*

4 Removes the tab page at the specified index from the collection. The 0-based index of the **System.Windows.Forms.TabPage** to remove.

5 *r) ICollection.CopyTo*

7 [C#] void ICollection.CopyTo(Array dest, int index);

8 [C++] void ICollection::CopyTo(Array* dest, int index);

9 [VB] Sub CopyTo(ByVal dest As Array, ByVal index As Integer) Implements
10 ICollection.CopyTo

11 [JScript] function ICollection.CopyTo(dest : Array, index : int);

12 *s) IList.Add*

14 [C#] int IList.Add(object value);

15 [C++] int IList::Add(Object* value);

16 [VB] Function Add(ByVal value As Object) As Integer Implements IList.Add

17 [JScript] function IList.Add(value : Object) : int;

18 *t) IList.Contains*

20 [C#] bool IList.Contains(object page);

21 [C++] bool IList::Contains(Object* page);

22 [VB] Function Contains(ByVal page As Object) As Boolean Implements
23 IList.Contains

24 [JScript] function IList.Contains(page : Object) : Boolean;

u) *IList.IndexOf*

[C#] int IList.IndexOf(object page);
[C++] int IList::IndexOf(Object* page);
[VB] Function IndexOf(ByVal page As Object) As Integer Implements
IList.IndexOf
[JScript] function IList.IndexOf(page : Object) : int;

v) *IList.Insert*

[C#] void IList.Insert(int index, object value);
[C++] void IList::Insert(int index, Object* value);
[VB] Sub Insert(ByVal index As Integer, ByVal value As Object) Implements
IList.Insert
[JScript] function IList.Insert(index : int, value : Object);

w) *IList.Remove*

[C#] void IList.Remove(object value);
[C++] void IList::Remove(Object* value);
[VB] Sub Remove(ByVal value As Object) Implements IList.Remove
[JScript] function IList.Remove(value : Object);

TabPage.TabPageControlCollection class (System.Windows.Forms)

a) *ToString*

Description

Contains the collection of controls that the **System.Windows.Forms.TabPage** uses.

b) *TabPage.TabPageControlCollection*

Example Syntax:

c) *ToString*

[C#] public TabPage.TabPageControlCollection(TabPage owner);

[C++] public: TabPageControlCollection(TabPage* owner);

[VB] Public Sub New(ByVal owner As TabPage)

[JScript] public function TabPage.TabPageControlCollection(owner : TabPage);

Description

Initializes a new instance of the **System.Windows.Forms.TabPage.TabPageControlCollection** class. The **System.Windows.Forms.TabPage** that contains this collection of controls.

d) *Count*

e) *IsReadOnly*

f) *Item*

g) *Add*

[C#] public override void Add(Control value);

[C++] public: void Add(Control* value);

[VB] Overrides Public Sub Add(ByVal value As Control)

[JScript] public override function Add(value : Control);

Description

Adds a control to the collection.

The specified control is added to the end of the collection. If the control is already a child of another control, it is removed from the other control. The control to add.

TabSizeMode enumeration (System.Windows.Forms)

a) *ToString*

Description

Specifies how tabs in a tab control are sized.

This enumeration is used by members such as
System.Windows.Forms.TabControl.SizeMode .

b) *ToString*

[C#]	public	const	TabSizeMode	FillToRight;
[C++]	public:	const	TabSizeMode	FillToRight;
[VB]	Public	Const	FillToRight	As TabSizeMode
[JScript]	public	var	FillToRight	: TabSizeMode;

Description

The width of each tab is sized so that each row of tabs fills the entire width of the container control. This is only applicable to tab controls with more than one row.

c) *ToString*

[C#]	public	const	TabSizeMode	Fixed;
[C++]	public:	const	TabSizeMode	Fixed;

[VB]	Public	Const	Fixed	As	TabSizeMode
[JScript]	public	var	Fixed	:	TabSizeMode;

Description

All tabs in a control are the same width.

d) ToString

[C#]	public	const	TabSizeMode	Normal;
[C++]	public:	const	TabSizeMode	Normal;
[VB]	Public	Const	Normal	As TabSizeMode
[JScript]	public	var	Normal	: TabSizeMode;

Description

The width of each tab is sized to accommodate what is displayed on the tab, and the size of tabs in a row are not adjusted to fill the entire width of the container control.

TextBox class (System.Windows.Forms)

a) ToString

Description

Represents a Windows text box control.

The **System.Windows.Forms.TextBox** control allows the user to enter text in an application. This control has additional functionality that is not found in the standard Windows text box control, including multiline editing and password character masking.

b) *TextBox*

Example Syntax:

c) *ToString*

[C#]	public	TextBox();
[C++]	public:	TextBox();
[VB]	Public Sub	New()
[JScript]	public function	TextBox();

Description

Initializes a new instance of the **System.Windows.Forms.TextBox** class.

The parent container control defines the color and font settings for the **System.Windows.Forms.TextBox** .

d) *AcceptsReturn*

e) *ToString*

[C#]	public	bool	AcceptsReturn	{get; set;}
[C++]	public: __property	bool	get_AcceptsReturn();	public: __property void
			set_AcceptsReturn(bool);	
[VB]	Public	Property	AcceptsReturn	As Boolean
[JScript]	public function	get	AcceptsReturn() : Boolean;	public function
			set	
			AcceptsReturn(Boolean);	

Description

Gets or sets a value indicating whether pressing ENTER in a multiline **System.Windows.Forms.TextBox** control creates a new line of text in the control or activates the default button for the form.

If the value of this property is **false** , the user must press CTRL+ENTER to create a new line in a multiline **System.Windows.Forms.TextBox** control. If there is no default button for the form, then the ENTER key will always create a new line of text in the control, no matter what the value of this property.

- f) AcceptsTab*
- g) AccessibilityObject*
- h) AccessibleDefaultActionDescription*
- i) AccessibleDescription*
- j) AccessibleName*
- k) AccessibleRole*
- l) AllowDrop*
- m) Anchor*
- n) AutoSize*
- o) BackColor*
- p) BackgroundImage*
- q) BindingContext*
- r) BorderStyle*
- s) Bottom*
- t) Bounds*
- u) CanFocus*
- v) CanSelect*
- w) CanUndo*
- x) Capture*
- y) CausesValidation*
- z) CharacterCasing*
- aa) ToString*

1
2
3 *Description*

4 Gets or sets whether the **System.Windows.Forms.TextBox** control modifies
5 the case of characters as they are typed.

6 You can use the **System.Windows.Forms.TextBox.CharacterCasing**
7 property to change the case of characters as required by your application. For
8 example, you could change the case of all characters entered in a
9 **System.Windows.Forms.TextBox** control used for password entry to
10 uppercase or lowercase to enforce a policy for passwords.

11 *bb) ClientRectangle*

12 *cc) ClientSize*

13 *dd) CompanyName*

14 *ee) Container*

15 *ff) ContainsFocus*

16 *gg) ContextMenu*

17 *hh) Controls*

18 *ii) Created*

19 *jj) CreateParams*

20 *kk) ToString*

21
22 *Description*

23 Returns the parameters needed to create the handle. Inheriting classes can
24 override this to provide extra functionality. They should not, however, forget to
25 call `base.CreateParams()` first to get the struct filled up with the basic info.

- ll) *Cursor*
- mm) *DataBindings*
- nn) *DefaultImeMode*
- oo) *ToString*

Description

1 *pp) DefaultSize*
2 *qq) DesignMode*
3 *rr) DisplayRectangle*
4 *ss) Disposing*
5 *tt) Dock*
6 *uu) Enabled*
7 *vv) Events*
8 *ww) Focused*
9 *xx) Font*
10 *yy) FontHeight*
11 *zz) ForeColor*
12 *aaa) Handle*
13 *bbb) HasChildren*
14 *ccc) Height*
15 *ddd) HideSelection*
16 *eee) ImeMode*
17 *fff) InvokeRequired*
18 *ggg) IsAccessible*
19 *hhh) IsDisposed*
20 *iii) IsHandleCreated*
21 *jjj) Left*
22 *kkk) Lines*
23 *lll) Location*

mmm) *MaxLength*

nnn) *Modified*

ooo) *Multiline*

ppp) *Name*

qqq) *Parent*

rrr) *PasswordChar*

sss) *ToString*

Description

Gets or sets the character used to mask characters of a password in a single-line **System.Windows.Forms.TextBox** control.

If the **System.Windows.Forms.TextBoxBase.Multiline** property is set to **true** , setting the **System.Windows.Forms.TextBox.PasswordChar** property has no visual effect. When the **System.Windows.Forms.TextBox.PasswordChar** property is set to **true** , cut, copy, and paste actions in the control using the keyboard are not allowed, regardless of whether the **System.Windows.Forms.TextBoxBase.Multiline** property is set to **true** or **false** .

1 *ttt) PreferredHeight*
2 *uuu) ProductName*
3 *vvv) ProductVersion*
4 *www) ReadOnly*
5 *xxx) RecreatingHandle*
6 *yyy) Region*
7 *zzz) RenderRightToLeft*
8 *aaaa) ResizeRedraw*
9 *bbbb) Right*
10 *cccc) RightToLeft*
11 *dddd) ScrollBars*
12 *eeee) ToString*

16 *Description*

17 Gets or sets which scroll bars should appear in a multiline
18 **System.Windows.Forms.TextBox** control.

19 Horizontal scroll bars will not be shown if the
20 **System.Windows.Forms.TextBoxBase.WordWrap** property is set to **true** ,
21 regardless of the value of the **System.Windows.Forms.TextBox.ScrollBars**
22 property.
23
24
25

ffff) SelectedText
gggg) SelectionLength
hhhh) SelectionStart
iiii) ShowFocusCues
jjjj) ShowKeyboardCues
kkkk) Site
llll) Size
mmmm) TabIndex
nnnn) TabStop
oooo) Tag
pppp) Text
qqqq) TextAlign
rrrr) ToString

Description

Gets or sets how text is aligned in a **System.Windows.Forms.TextBox** control.

You can use this property to align the text within a **System.Windows.Forms.TextBox** to match the layout of text on your form. For example, if your controls are all located on the right side of the form, you can set the **System.Windows.Forms.Label.TextAlign** property to **HorizontalAlignment.Right**, and the text will be aligned with the right side of the control instead of the default left alignment.

1 *ssss) TextLength*

2 *tttt) Top*

3 *uuuu) TopLevelControl*

4 *vvvv) Visible*

5 *www) Width*

6 *xxxx) WindowTarget*

7 *yyyy) WordWrap*

8 *zzzz) ToString*

9
10
11 *Description*

12 Occurs when the value of the **System.Windows.Forms.TextBox.TextAlign**
13 property has changed.

14 For more information about handling events, see .

15 *aaaaa) IsInputKey*

16
17 [C#] protected override bool IsInputKey(Keys keyData);

18 [C++] protected: bool IsInputKey(Keys keyData);

19 [VB] Overrides Protected Function IsInputKey(ByVal keyData As Keys) As
20 Boolean

21 [JScript] protected override function IsInputKey(keyData : Keys) : Boolean;

22
23 *Description*

24 Overridden to handle RETURN key.

bbbb) OnHandleCreated

```
1  
2 [C#]    protected    override    void    OnHandleCreated(EventArgs    e);  
3 [C++]    protected:    void    OnHandleCreated(EventArgs*    e);  
4 [VB] Overrides Protected Sub OnHandleCreated(ByVal e As EventArgs)  
5 [JScript] protected override function OnHandleCreated(e : EventArgs);  
6
```

Description

Overridden to update the newly created handle with the settings of the PasswordChar properties.

cccc) OnMouseUp

```
11  
12 [C#]    protected    override    void    OnMouseUp(MouseEventArgs    mevent);  
13 [C++]    protected:    void    OnMouseUp(MouseEventArgs*    mevent);  
14 [VB] Overrides Protected Sub OnMouseUp(ByVal mevent As MouseEventArgs)  
15 [JScript] protected override function OnMouseUp(mevent : MouseEventArgs);  
16 Raises                                     the  
17 System.Windows.Forms.TextBox.OnMouseUp(System.Windows.Forms.Mous  
18 eEventArgs) event.
```

dddd) OnTextAlignChanged

```
21 [C#]    protected    virtual    void    OnTextAlignChanged(EventArgs    e);  
22 [C++]    protected:    virtual    void    OnTextAlignChanged(EventArgs*    e);  
23 [VB] Overridable Protected Sub OnTextAlignChanged(ByVal e As EventArgs)  
24 [JScript]    protected    function    OnTextAlignChanged(e : EventArgs);  
25
```

Description

Raises the **System.Windows.Forms.TextBox.TextAlignChanged** event.

Raising an event invokes the event handler through a delegate. For more information, see . An **System.EventArgs** that contains the event data.

eeeeee) WndProc

[C#] protected override void WndProc(ref Message m);

[C++] protected: void WndProc(Message* m);

[VB] Overrides Protected Sub WndProc(ByRef m As Message)

[JScript] protected override function WndProc(m : Message);

Description

The edits window procedure. Inheriting classes can override this to add extra functionality, but should not forget to call `base.wndProc(m)`; to ensure the combo continues to function properly. A Windows Message Object.

TextBoxBase class (System.Windows.Forms)

a) WndProc

Description

Implements the basic functionality required by text controls.

This class implements the core features of text manipulation controls, such as **System.Windows.Forms.TextBox** and **System.Windows.Forms.RichTextBox** . These include text selection, Clipboard functionality, multiline text control support, and many events.

b) *AcceptsTab*

c) *WndProc*

[C#] public bool AcceptsTab {get; set;}

[C++] public: __property bool get_AcceptsTab();public: __property void
set_AcceptsTab(bool);

[VB] Public Property AcceptsTab As Boolean

[JScript] public function get AcceptsTab() : Boolean;public function set
AcceptsTab(Boolean);

Description

Gets or sets a value indicating whether pressing the TAB key in a multiline text box control types a TAB character in the control instead of moving the focus to the next control in the tab order.

If the **System.Windows.Forms.TextBoxBase.AcceptsTab** property is set to **true** , the user must press CTRL+TAB to move the focus to the next control in the tab order.

- d) *AccessibilityObject*
- e) *AccessibleDefaultActionDescription*
- f) *AccessibleDescription*
- g) *AccessibleName*
- h) *AccessibleRole*
- i) *AllowDrop*
- j) *Anchor*
- k) *AutoSize*
- l) *WndProc*

Description

Gets or sets a value indicating whether the height of the control automatically adjusts when the font assigned to the control is changed.

Multiline text box controls ignore this property setting. When this property is set to **true** , the text box control resizes its height based on the size of the font assigned to the control. You can use this property to ensure that the user can read text assigned to a single-line version of the control regardless of the font.

m) *BackColor*

n) *WndProc*

```
[C#]      public      override      Color      BackColor      {get;      set;}
```

```
[C++] public: __property virtual Color get_BackColor();public: __property virtual  
void                                           set_BackColor(Color);
```

```
[VB]      Overrides      Public      Property      BackColor      As      Color
```

```
[JScript] public function get BackColor() : Color;public function set
```


BackColor(Color);

Description

Gets or sets the background color of the control.

You can use the **System.Windows.Forms.TextBoxBase.BackColor** property to change the background color of the text control to blend into the color scheme of your forms.

o) BackgroundImage

p) WndProc

[C#] public override Image BackgroundImage {get; set;}

[C++] public: __property virtual Image* get_BackgroundImage();public:

__property virtual void set_BackgroundImage(Image*);

[VB] Overrides Public Property BackgroundImage As Image

[JScript] public function get BackgroundImage() : Image;public function set

BackgroundImage(Image);

Description

q) BindingContext

r) BorderStyle

s) WndProc

Description

Gets or sets the border type of the text box control.

1 You can use the **System.Windows.Forms.TextBoxBase.BorderStyle**
2 property to create borderless and flat style controls, in addition to the default
3 three-dimensional control.

4 *t) Bottom*

5 *u) Bounds*

6 *v) CanFocus*

7 *w) CanSelect*

8 *x) CanUndo*

9 *y) WndProc*

10 *Description*

11 Gets a value indicating whether the user can undo the previous operation in a
12 text box control.

13 If this method returns **true** , you can call the
14 **System.Windows.Forms.TextBoxBase.Undo** method to undo the last
15 operation in a text box. You can use this method in the
16 **System.Windows.Forms.MenuItem.Popup** event of a
17 **System.Windows.Forms.MenuItem** , or in code that manages the state of
18 buttons on a **System.Windows.Forms.ToolBar** to enable or disable the ability
19 to undo the previous operation in a text box control.
20
21
22
23
24
25

- z) *Capture*
- aa) *CausesValidation*
- bb) *ClientRectangle*
- cc) *ClientSize*
- dd) *CompanyName*
- ee) *Container*
- ff) *ContainsFocus*
- gg) *ContextMenu*
- hh) *Controls*
- ii) *Created*
- jj) *CreateParams*
- kk) *WndProc*

Description

Returns the parameters needed to create the handle. Inheriting classes can override this to provide extra functionality. They should not, however, forget to call `base.CreateParams()` first to get the struct filled up with the basic info.

1 **ll) Cursor**

2 **mm) DataBindings**

3 **nn) DefaultImeMode**

4 **oo) DefaultSize**

5 **pp) WndProc**

6
7
8 *Description*

9 Deriving classes can override this to configure a default size for their control.
10 This is more efficient than setting the size in the control's constructor.

11 **qq) DesignMode**

12 **rr) DisplayRectangle**

13 **ss) Disposing**

14 **tt) Dock**

15 **uu) Enabled**

16 **vv) Events**

17 **ww) Focused**

18 **xx) Font**

19 **yy) FontHeight**

20 **zz) ForeColor**

21 **aaa) WndProc**

22
23
24
25 *Description*

Gets or sets the foreground color of the control.

You can use the **System.Windows.Forms.TextBoxBase.ForeColor** property to change the color of the text within the control to match the text of other controls on your form. You can also use this property to highlight a specific text box that contains an invalid value.

bbb) Handle

ccc) HasChildren

ddd) Height

eee) HideSelection

fff) WndProc

Description

Gets or sets a value indicating whether the selected text in the text box control remains highlighted when the control loses focus.

You can use this property to keep text highlighted in a text box control while another form or a dialog box has focus, such as a spelling checker dialog box.

1 *ggg) ImeMode*

2 *hhh) InvokeRequired*

3 *iii) IsAccessible*

4 *jjj) IsDisposed*

5 *kkk) IsHandleCreated*

6 *lll) Left*

7 *mmm) Lines*

8 *nnn) WndProc*

9
10
11 *Description*

12 Gets or sets the lines of text in a text box control.

13 Each element in the array becomes a line of text in the text box control. If the
14 **System.Windows.Forms.TextBoxBase.Multiline** property of the text box
15 control is set to **true** and a newline character appears in the text, the text
16 following the newline character is added to a new element in the array and
17 displayed on a separate line.

16 *ooo) Location*

17 *ppp) MaxLength*

18 *qqq) WndProc*

19
20
21
22 *Description*

23 Gets or sets the maximum number of characters the user can type into the text
24 box control.

24 When this property is set to zero, the maximum length of the text that can be
25 entered in the control is limited only by available memory. You can use this
26 property to restrict the length of text entered in the control for values such as

[VB] Overridable Public Property Multiline As Boolean
[JScript] public function get Multiline() : Boolean; public function set
Multiline(Boolean);

Description

Gets or sets a value indicating whether this is a multiline text box control.

A multiline text box allows you to display more than one line of text in the control. If the **System.Windows.Forms.TextBoxBase.WordWrap** property is set to **true**, text entered into the multiline text box is wrapped to the next line in the control. If the **System.Windows.Forms.TextBoxBase.WordWrap** property is set to **false**, text entered into the multiline text box control will be displayed on the same line until a newline character is entered.

vvv) *Name*

www) *Parent*

xxx) *PreferredHeight*

yyy) *WndProc*

Description

Gets the preferred height for a single-line text box.

The size returned by this property is based on the font height and border style of the text box. You can use this property to determine the appropriate size of the text box to ensure that text is properly displayed in the control. The value returned by this property is in pixels.

1 *zzz) **ProductName***

2 *aaaa) **ProductVersion***

3 *bbbb) **ReadOnly***

4 *cccc) **WndProc***

5
6
7 *Description*

8 Gets or sets a value indicating whether text in the text box is read-only.

9 When this property is set to **true** , the contents of the control cannot be
10 changed by the user at runtime. With this property set to **true** , you can still set
11 the value of the **System.Windows.Forms.TextBoxBase.Text** property in
12 code. You can use this feature instead of disabling the control with the
13 **System.Windows.Forms.Control.Enabled** property to allow the contents to
14 be copied.

15 *dddd) **RecreatingHandle***

16 *eeee) **Region***

17 *ffff) **RenderRightToLeft***

18 *gggg) **ResizeRedraw***

19 *hhhh) **Right***

20 *iiii) **RightToLeft***

21 *jjjj) **SelectedText***

22 *kkkk) **WndProc***

23 *Description*

24 Gets or sets a value indicating the currently selected text in the control.

You can assign text to this property to change the text currently selected in the text box. If no text is currently selected in the text box, this property returns a zero-length string.

III) SelectionLength

mmmm)WndProc

```
[C#]      public      virtual      int      SelectionLength      {get;      set;}
[C++] public: __property virtual int get_SelectionLength();public: __property
virtual                                     void                                     set_SelectionLength(int);
[VB]      Overridable      Public      Property      SelectionLength      As      Integer
[JScript] public function get SelectionLength() : int;public function set
SelectionLength(int);
```

Description

Gets or sets the number of characters selected in the text box.

You can use this property to determine if any characters are currently selected in the text box control before performing operations on the selected text. When the value of the **System.Windows.Forms.TextBoxBase.SelectionLength** property is set to a value that is larger than the number of characters within the text of the control, the value of the **System.Windows.Forms.TextBoxBase.SelectionLength** property is set to the entire length of text within the control minus the value of the **System.Windows.Forms.TextBoxBase.SelectionStart** property (if any value is specified for the **System.Windows.Forms.TextBoxBase.SelectionStart** property).

nnnn) SelectionStart

oooo) WndProc

```
[C#]      public      int      SelectionStart      {get;      set;}
[C++] public: __property int get_SelectionStart();public: __property void
```

1 set_SelectionStart(int);

2 [VB] Public Property SelectionStart As Integer

3 [JScript] public function get SelectionStart() : int; public function set

4 SelectionStart(int);

5
6 *Description*

7 Gets or sets the starting point of text selected in the text box.

8 If no text is selected in the control, this property indicates the insertion point for
9 new text. If you set this property to a location beyond the length of the text in
the control, the selection start position will be placed after the last character.

10 When text is selected in the text box control, changing this property may
decrease the value of the

11 **System.Windows.Forms.TextBoxBase.SelectionLength** property. If the
remaining text in the control after the position indicated by the
12 **System.Windows.Forms.TextBoxBase.SelectionStart** property is less than
the value of the **System.Windows.Forms.TextBoxBase.SelectionLength**
property, the value of the

13 **System.Windows.Forms.TextBoxBase.SelectionLength** property is
automatically decreased. The value of the

14 **System.Windows.Forms.TextBoxBase.SelectionStart** property never
causes an increase in the

15 **System.Windows.Forms.TextBoxBase.SelectionLength** property.
16
17
18
19
20
21
22
23
24
25

pppp) *ShowFocusCues*
 qqqq) *ShowKeyboardCues*
 rrrr) *Site*
 ssss) *Size*
 tttt) *TabIndex*
 uuuu) *TabStop*
 vvvv) *Tag*
 wwww) *Text*
 xxxx) *WndProc*

Description

Gets or sets the current text in the text box.

To display multiple lines of text in a text box, set the **System.Windows.Forms.TextBoxBase.Multiline** property to **true** . To read or set the text of a multiline text box, use the **System.Windows.Forms.TextBoxBase.Lines** property. The amount of text that can be stored in the **System.Windows.Forms.TextBoxBase.Text** property is limited to 65KB of memory for the **System.Windows.Forms.TextBox** control. The amount of text that can be entered in the **System.Windows.Forms.RichTextBox** control is limited only by available system memory.

yyyy) *TextLength*
 zzzz) *WndProc*

[C#]	public	int	TextLength	{get;}
[C++]	public:	__property	int	get_TextLength();
[VB]	Public	ReadOnly	Property	TextLength As Integer

1 [JScript] public function get TextLength() : int;

3 *Description*

4 Gets the length of text in the control.

5 You can use this property to determine the number of characters in a string for
6 tasks such as searching for specific strings of text within the text of the control,
where knowledge of the total number of characters is needed.

7 *aaaaa) Top*

8 *bbbbbb) TopLevelControl*

9 *ccccc) Visible*

10 *dddddd) Width*

11 *eeeeee) WindowTarget*

12 *ffffff) WordWrap*

13 *ggggg) WndProc*

16
17 *Description*

18 Indicates whether a multiline text box control automatically wraps words to the
beginning of the next line when necessary.

19 If this property is set to **true** , horizontal scroll bars are not displayed regardless
20 of the **System.Windows.Forms.TextBox.ScrollBars** property setting.

21 *hhhhh)WndProc*

23 [C#] public event EventHandler AcceptsTabChanged;

24 [C++] public: __event EventHandler* AcceptsTabChanged;

25 [VB] Public Event AcceptsTabChanged As EventHandler

Description

Occurs when the value of the
System.Windows.Forms.TextBoxBase.AcceptsTab property has changed.

For more information about handling events, see .

iiii) WndProc

[C#] public event EventHandler AutoSizeChanged;

[C++] public: __event EventHandler* AutoSizeChanged;

[VB] Public Event AutoSizeChanged As EventHandler

Description

Occurs when the value of the
System.Windows.Forms.TextBoxBase.AutoSize property has changed.

For more information about handling events, see .

jjjj) WndProc

Description

Occurs when the value of the
System.Windows.Forms.TextBoxBase.BorderStyle property has changed.

For more information about handling events, see .

kkkkk) WndProc

Description

Occurs when the text box is clicked.

For more information about handling events, see .

lllll) WndProc

Description

Occurs when the value of the
System.Windows.Forms.TextBoxBase.HideSelection property has
changed.

For more information about handling events, see .

mmmmm)WndProc

Description

Occurs when the value of the
System.Windows.Forms.TextBoxBase.Modified property has changed.

For more information about handling events, see .

nnnnn)WndProc

Description

Occurs when the value of the
System.Windows.Forms.TextBoxBase.Multiline property has changed.

For more information about handling events, see .

ooooo) WndProc

Description

Occurs when the value of the **System.Windows.Forms.TextBoxBase.ReadOnly** property has changed.

For more information about handling events, see .

ppppp) AppendText

[C#]	public	void	AppendText(string	text);
[C++]	public:	void	AppendText(String*	text);
[VB]	Public	Sub	AppendText(ByVal	text As String)
[JScript]	public	function	AppendText(text	: String);

Description

Appends text to the current text of text box.

You can use this method to add text to the existing text in the control instead of using the concatenation operator (+) to concatenate text to the **System.Windows.Forms.TextBoxBase.Text** property. The text to append to the current contents of the text box.

qqqqq) Clear

[C#]	public	void	Clear();
[C++]	public:	void	Clear();
[VB]	Public	Sub	Clear()
[JScript]	public	function	Clear();

Description

Clears all text from the text box control.

You can use this method to clear the contents of the control instead of assigning the **System.Windows.Forms.TextBoxBase.Text** property an empty string.

rrrrr) ClearUndo

[C#]	public	void	ClearUndo();
[C++]	public:	void	ClearUndo();
[VB]	Public	Sub	ClearUndo()
[JScript]	public	function	ClearUndo();

Description

Clears information about the most recent operation from the undo buffer of the text box.

You can use this method to prevent an undo operation from repeating, based on the state of your application.

sssss) Copy

[C#]	public	void	Copy();
[C++]	public:	void	Copy();
[VB]	Public	Sub	Copy()
[JScript]	public	function	Copy();

Description

Copies the current selection in the text box to the Clipboard.

You can use this method, instead of using the **System.Windows.Forms.Clipboard** class, to copy text in the text box and place it in the Clipboard.

tttt) CreateHandle

[C#]	protected	override	void	CreateHandle();
[C++]	protected:		void	CreateHandle();
[VB]	Overrides	Protected	Sub	CreateHandle()
[JScript]	protected	override	function	CreateHandle();

Description

uuuuu)Cut

[C#]	public	void	Cut();
[C++]	public:	void	Cut();
[VB]	Public	Sub	Cut()
[JScript]	public	function	Cut();

Description

Moves the current selection in the text box to the Clipboard.

This method will only cut text from the text box if text is selected in the control. You can use this method, instead of using the **System.Windows.Forms.Clipboard** class, to copy text in the text box and move it to the Clipboard.

vvvvv) IsInputKey

[C#]	protected	override	bool	IsInputKey(Keys	keyData);
------	-----------	----------	------	-----------------	-----------

1 [C++] protected: bool IsInputKey(Keys keyData);

2 [VB] Overrides Protected Function IsInputKey(ByVal keyData As Keys) As
3 Boolean

4 [JScript] protected override function IsInputKey(keyData : Keys) : Boolean;

6 *Description*

7 Overridden to handle TAB key.

8 *www)OnAcceptsTabChanged*

10 [C#] protected virtual void OnAcceptsTabChanged(EventArgs e);

11 [C++] protected: virtual void OnAcceptsTabChanged(EventArgs* e);

12 [VB] Overridable Protected Sub OnAcceptsTabChanged(ByVal e As EventArgs)

13 [JScript] protected function OnAcceptsTabChanged(e : EventArgs);

15 *Description*

16 Raises the **System.Windows.Forms.TextBoxBase.AcceptsTabChanged**
event.

17 Raising an event invokes the event handler through a delegate. For more
18 information, see . An **System.EventArgs** that contains the event data.

19 *xxxxx) OnAutoSizeChanged*

21 [C#] protected virtual void OnAutoSizeChanged(EventArgs e);

22 [C++] protected: virtual void OnAutoSizeChanged(EventArgs* e);

23 [VB] Overridable Protected Sub OnAutoSizeChanged(ByVal e As EventArgs)

24 [JScript] protected function OnAutoSizeChanged(e : EventArgs);

Description

Raises the **System.Windows.Forms.TextBoxBase.AutoSizeChanged** event.

Raising an event invokes the event handler through a delegate. For more information, see . An **System.EventArgs** that contains the event data.

yyyyy) OnBorderStyleChanged

[C#] protected virtual void OnBorderStyleChanged(EventArgs e);

[C++] protected: virtual void OnBorderStyleChanged(EventArgs* e);

[VB] Overridable Protected Sub OnBorderStyleChanged(ByVal e As EventArgs)

[JScript] protected function OnBorderStyleChanged(e : EventArgs);

Description

Raises the **System.Windows.Forms.TextBoxBase.BorderStyleChanged** event.

Raising an event invokes the event handler through a delegate. For more information, see . An **System.EventArgs** that contains the event data.

zzzzz) OnFontChanged

[C#] protected override void OnFontChanged(EventArgs e);

[C++] protected: void OnFontChanged(EventArgs* e);

[VB] Overrides Protected Sub OnFontChanged(ByVal e As EventArgs)

[JScript] protected override function OnFontChanged(e : EventArgs);

Description

aaaaaa)OnHandleCreated

```
1  
2  
3 [C#]    protected    override    void    OnHandleCreated(EventArgs    e);  
4 [C++]    protected:    void    OnHandleCreated(EventArgs*    e);  
5 [VB] Overrides Protected Sub OnHandleCreated(ByVal e As EventArgs)  
6 [JScript] protected override function OnHandleCreated(e : EventArgs);
```

Description

Overridden to update the newly created handle with the settings of the MaxLength and PasswordChar properties.

bbbbbb)OnHandleDestroyed

```
12 [C#]    protected    override    void    OnHandleDestroyed(EventArgs    e);  
13 [C++]    protected:    void    OnHandleDestroyed(EventArgs*    e);  
14 [VB] Overrides Protected Sub OnHandleDestroyed(ByVal e As EventArgs)  
15 [JScript] protected override function OnHandleDestroyed(e : EventArgs);
```

Description

cccccc)OnHideSelectionChanged

```
20 [C#]    protected    virtual    void    OnHideSelectionChanged(EventArgs    e);  
21 [C++]    protected:    virtual    void    OnHideSelectionChanged(EventArgs*    e);  
22 [VB] Overridable Protected Sub OnHideSelectionChanged(ByVal e As  
23 EventArgs)  
24 [JScript] protected function OnHideSelectionChanged(e : EventArgs);  
25
```

1
2 *Description*

3 Raise the **System.Windows.Forms.TextBoxBase.HideSelectionChanged**
4 event.

5 Raising an event invokes the event handler through a delegate. For more
6 information, see . An **System.EventArgs** that contains the event data.

7
8 *dddddd)OnModifiedChanged*

9 [C#] protected virtual void OnModifiedChanged(EventArgs e);

10 [C++] protected: virtual void OnModifiedChanged(EventArgs* e);

11 [VB] Overridable Protected Sub OnModifiedChanged(ByVal e As EventArgs)

12 [JScript] protected function OnModifiedChanged(e : EventArgs);

13 *Description*

14 Raises the **System.Windows.Forms.TextBoxBase.ModifiedChanged** event.

15 Raising an event invokes the event handler through a delegate. For more
16 information, see . An **System.EventArgs** that contains the event data.

17 *eeeeee)OnMultilineChanged*

18 [C#] protected virtual void OnMultilineChanged(EventArgs e);

19 [C++] protected: virtual void OnMultilineChanged(EventArgs* e);

20 [VB] Overridable Protected Sub OnMultilineChanged(ByVal e As EventArgs)

21 [JScript] protected function OnMultilineChanged(e : EventArgs);

22
23 *Description*

24 Raises the **System.Windows.Forms.TextBoxBase.MultilineChanged** event.

Raising an event invokes the event handler through a delegate. For more information, see . An **System.EventArgs** that contains the event data.

fffff) OnReadOnlyChanged

[C#] protected virtual void OnReadOnlyChanged(EventArgs e);

[C++] protected: virtual void OnReadOnlyChanged(EventArgs* e);

[VB] Overridable Protected Sub OnReadOnlyChanged(ByVal e As EventArgs)

[JScript] protected function OnReadOnlyChanged(e : EventArgs);

Description

Raises the **System.Windows.Forms.TextBoxBase.ReadOnlyChanged** event.

Raising an event invokes the event handler through a delegate. For more information, see . An **System.EventArgs** that contains the event data.

ggggg) Paste

[C#] public void Paste();

[C++] public: void Paste();

[VB] Public Sub Paste()

[JScript] public function Paste();

Description

Replaces the current selection in the text box with the contents of the Clipboard.

The **System.Windows.Forms.TextBoxBase.Paste** method will only paste text into the control if text is currently stored in the Clipboard. Once your application exits, any content stored in the Clipboard is removed.

hhhhh)ProcessDialogKey

```
1
2
3 [C#]    protected    override    bool    ProcessDialogKey(Keys    keyData);
4
5 [C++]    protected:    bool    ProcessDialogKey(Keys    keyData);
6
7 [VB] Overrides Protected Function ProcessDialogKey(ByVal keyData As Keys)
8 As Boolean
9
10 [JScript] protected override function ProcessDialogKey(keyData : Keys) :
11 Boolean;
```

iiiiii) ScrollToCaret

```
10
11 [C#]                public                void                ScrollToCaret();
12
13 [C++]                public:                void                ScrollToCaret();
14
15 [VB]                Public                Sub                ScrollToCaret()
16
17 [JScript]                public                function                ScrollToCaret();
```

Description

Scrolls the contents of the control to the current caret position.

This method enables you to scroll the contents of the control until the caret is within the visible region of the control. If the caret is positioned below the visible region of the control, the

System.Windows.Forms.TextBoxBase.ScrollToCaret method will scroll the contents of the control until the caret is visible at the bottom of the control. If the caret is positioned above the visible region of the control, this method scrolls the contents of the control until the caret is visible at the top of the control. You can use this method in a multiline text box to ensure that the current text entry point is within the visible region of the control.

jjjjj) Select

[C#] public void Select(int start, int length);

[C++] public: void Select(int start, int length);

[VB] Public Sub Select(ByVal start As Integer, ByVal length As Integer)

[JScript] public function Select(start : int, length : int); Selects text within the control.

Description

Selects a range of text in the text box.

If you want to set the start position to the first character in the control's text, set the *start* parameter to 0. You can use this method to select a substring of text, such as when searching through the text of the control and replacing information. The position of the first character in the current text selection within the text box. The number of characters to select.

kkkkkk)SelectAll

[C#] public void SelectAll();

[C++] public: void SelectAll();

[VB] Public Sub SelectAll()

[JScript] public function SelectAll();

Description

Selects all text in the text box.

This method enables you to select all text within the control. You can use this method in conjunction with the **System.Windows.Forms.TextBoxBase.Cut** method, which requires text to be selected in the control, to cut the entire contents of the control and paste them into the Clipboard.

IIIIII) SetBoundsCore

[C#] protected override void SetBoundsCore(int x, int y, int width, int height, BoundsSpecified specified);

[C++] protected: void SetBoundsCore(int x, int y, int width, int height, BoundsSpecified specified);

[VB] Overrides Protected Sub SetBoundsCore(ByVal x As Integer, ByVal y As Integer, ByVal width As Integer, ByVal height As Integer, ByVal specified As BoundsSpecified)

[JScript] protected override function SetBoundsCore(x : int, y : int, width : int, height : int, specified : BoundsSpecified);

Description

Overrides Control.setBoundsCore to enforce autoSize.

mmmmmm)ToString

[C#] public override string ToString();

[C++] public: String* ToString();

[VB] Overrides Public Function ToString() As String

[JScript] public override function ToString() : String;

Description

Provides some interesting information for the TextBox control in String form.

Return Value: String Provides some interesting information for the TextBox control in String form.

nnnnnn)Undo

[C#]	public	void	Undo();
[C++]	public:	void	Undo();
[VB]	Public	Sub	Undo()
[JScript]	public	function	Undo();

Description

Undoes the last edit operation in the text box.

This method will undo the last clipboard or text change operation performed in the text box control if the **System.Windows.Forms.TextBoxBase.CanUndo** property returns **true** .

oooooo)WndProc

[C#]	protected	override	void	WndProc(ref	Message	m);
[C++]	protected:		void	WndProc(Message*		m);
[VB]	Overrides	Protected	Sub	WndProc(ByRef	m	As Message)
[JScript]	protected	override	function	WndProc(m	:	Message);

Description

The edits window procedure. Inheriting classes can override this to add extra functionality, but should not forget to call `base.wndProc(m)`; to ensure the combo continues to function properly. A Windows Message Object.

ThreadExceptionHandler class (System.Windows.Forms)

a) *WndProc*

Description

Implements a dialog box that is displayed when an unhandled exception occurs in a thread.

b) *ThreadExceptionHandler*

Example Syntax:

c) *WndProc*

[C#] public ThreadExceptionHandler(Exception t);

[C++] public: ThreadExceptionHandler(Exception* t);

[VB] Public Sub New(ByVal t As Exception)

[JScript] public function ThreadExceptionHandler(t : Exception);

Description

Initializes a new instance of the **System.Windows.Forms.ThreadExceptionHandler** class.

Information contained in will be displayed in the **System.Windows.Forms.ThreadExceptionHandler** box. The **System.Exception** that represents the exception that occurred.

- d) *AcceptButton*
- e) *AccessibilityObject*
- f) *AccessibleDefaultActionDescription*
- g) *AccessibleDescription*
- h) *AccessibleName*
- i) *AccessibleRole*
- j) *ActiveControl*
- k) *ActiveMdiChild*
- l) *AllowDrop*
- m) *AllowTransparency*
- n) *Anchor*
- o) *AutoScale*
- p) *AutoScaleBaseSize*
- q) *AutoScroll*
- r) *AutoScrollMargin*
- s) *AutoScrollMinSize*
- t) *AutoScrollPosition*
- u) *BackColor*
- v) *BackgroundImage*
- w) *BindingContext*
- x) *Bottom*
- y) *Bounds*
- z) *CancelButton*

1	aa) CanFocus
2	bb) CanSelect
3	cc) Capture
4	dd) CausesValidation
5	ee) ClientRectangle
6	ff) ClientSize
7	gg) CompanyName
8	hh) Container
9	ii) ContainsFocus
10	jj) ContextMenu
11	kk) ControlBox
12	ll) Controls
13	mm) Created
14	nn) CreateParams
15	oo) Cursor
16	pp) DataBindings
17	qq) DefaultImeMode
18	rr) DefaultSize
19	ss) DesignMode
20	tt) DesktopBounds
21	uu) DesktopLocation
22	vv) DialogResult
23	ww) DisplayRectangle
24	
25	

1	xx) Disposing
2	yy) Dock
3	zz) DockPadding
4	aaa) Enabled
5	bbb) Events
6	ccc) Focused
7	ddd) Font
8	eee) FontHeight
9	fff) ForeColor
10	ggg) FormBorderStyle
11	hhh) Handle
12	iii) HasChildren
13	jjj) Height
14	kkk) HelpButton
15	lll) HScroll
16	mmm) Icon
17	nnn) ImeMode
18	ooo) InvokeRequired
19	ppp) IsAccessible
20	qqq) IsDisposed
21	rrr) IsHandleCreated
22	sss) IsMdiChild
23	ttt) IsMdiContainer
24	
25	

uuu) *IsRestrictedWindow*

vvv) *KeyPreview*

www) *Left*

xxx) *Location*

yyy) *MaximizeBox*

zzz) *MaximizedBounds*

aaaa) *MaximumSize*

bbbb) *MdiChildren*

cccc) *MdiParent*

dddd) *Menu*

eeee) *MergedMenu*

ffff) *MinimizeBox*

gggg) *MinimumSize*

hhhh) *Modal*

iii) *Name*

jjjj) *Opacity*

kkkk) *OwnedForms*

llll) *Owner*

mmmm) *Parent*

nnnn) *ParentForm*

oooo) *ProductName*

pppp) *ProductVersion*

qqqq) *RecreatingHandle*

rrrr) Region
ssss) RenderRightToLeft
tttt) ResizeRedraw
uuuu) Right
vvvv) RightToLeft
www) ShowFocusCues
xxxx) ShowInTaskbar
yyyy) ShowKeyboardCues
zzzz) Site
aaaaa) Size
bbbbbb) SizeGripStyle
cccc) StartPosition
dddd) TabIndex
eeee) TabStop
ffff) Tag
ggggg) Text
hhhhh) Top
iiii) TopLevel
jjjj) TopLevelControl
kkkkk) TopMost
llll) TransparencyKey
mmmm) Visible
nnnnn) VScroll

ooooo) Width

ppppp) WindowState

qqqqq) WindowTarget

TickStyle enumeration (System.Windows.Forms)

a) WndProc

Description

Specifies the location of tick marks in a **System.Windows.Forms.TrackBar** control.

Use the members of this enumeration to set the value of the **System.Windows.Forms.TrackBar.TickStyle** property of the **System.Windows.Forms.TrackBar** control.

b) WndProc

[C#]	public	const	TickStyle	Both;
[C++]	public:	const	TickStyle	Both;
[VB]	Public	Const	Both As	TickStyle
[JScript]	public	var	Both :	TickStyle;

Description

Tick marks are located on both sides of the control.

c) WndProc

[C#]	public	const	TickStyle	BottomRight;
[C++]	public:	const	TickStyle	BottomRight;

[VB]	Public	Const	BottomRight	As	TickStyle
[JScript]	public	var	BottomRight	:	TickStyle;

Description

Tick marks are located on the bottom of a horizontal control or on the right side of a vertical control.

d) WndProc

[C#]	public	const	TickStyle	None;	
[C++]	public:	const	TickStyle	None;	
[VB]	Public	Const	None	As	TickStyle
[JScript]	public	var	None	:	TickStyle;

Description

No tick marks appear in the control.

e) WndProc

[C#]	public	const	TickStyle	TopLeft;	
[C++]	public:	const	TickStyle	TopLeft;	
[VB]	Public	Const	TopLeft	As	TickStyle
[JScript]	public	var	TopLeft	:	TickStyle;

Description

Tick marks are located on the top of a horizontal control or on the left of a vertical control.

Timer class (System.Windows.Forms)

a) ToString

Description

Implements a timer that raises an event at user-defined intervals. This timer is optimized for use in Windows Forms applications and must be used in a window.

A **System.Windows.Forms.Timer** is used to raise an event at user-defined intervals. This Windows timer is designed for a single-threaded environment where UI threads are used to perform processing. It requires that the user code have a UI message pump available and always operate from the same thread, or marshal the call onto another thread.

b) Timer

Example Syntax:

c) ToString

[C#]	public	Timer();
[C++]	public:	Timer();
[VB]	Public	Sub New()
[JScript]	public function	Timer();

Initializes a new instance of the **System.Windows.Forms.Timer** class.

Description

Initializes a new instance of the **System.Windows.Forms.Timer** class.

When a new timer is created, it is disabled; that is, **System.Windows.Forms.Timer.Enabled** is set to **false** . To enable the timer, call the **System.Windows.Forms.Timer.Start** method or set **System.Windows.Forms.Timer.Enabled** to **true** .

d) *Timer*

Example Syntax:

e) *ToString*

```
[C#]          public          Timer(IContainer          container);
[C++]          public:          Timer(IContainer*          container);
[VB]   Public   Sub   New(ByVal   container   As   IContainer)
[JScript]   public   function   Timer(container   :   IContainer);
```

Description

Initializes a new instance of the **System.Windows.Forms.Timer** class with the specified container.

When a new timer is created, it is disabled; that is, **System.Windows.Forms.Timer.Enabled** is set to **false** . To enable the timer, call the **System.Windows.Forms.Timer.Start** method or set **System.Windows.Forms.Timer.Enabled** to **true** . An **System.ComponentModel.IContainer** that represents the container for the timer.

f) *Container*

g) *DesignMode*

h) *Enabled*

i) *ToString*

Description

Gets or sets whether the timer is running.

The timer is not subject to garbage collection when the value is **true** .

j) *Events*

k) *Interval*

l) *ToString*

Description

Gets or sets the time, in milliseconds, between timer ticks.

To get the number of seconds in the interval, divide this number by 1,000.

m) *Site*

n) *ToString*

Description

Occurs when the specified timer interval has elapsed and the timer is enabled.

For more information about handling events, see .

o) *Dispose*

[C#] protected override void Dispose(bool disposing);

[C++] protected: void Dispose(bool disposing);

[VB] Overrides Protected Sub Dispose(ByVal disposing As Boolean)

[JScript] protected override function Dispose(disposing : Boolean);

Description

Disposes of the resources (other than memory) used by the timer.

Call **System.Windows.Forms.Timer.Dispose(System.Boolean)** when you are finished using the timer. The

System.Windows.Forms.Timer.Dispose(System.Boolean) method leaves the timer in an unusable state. After calling **System.Windows.Forms.Timer.Dispose(System.Boolean)**, you must release all references to the timer so the memory it was occupying can be reclaimed by garbage collection.

p) OnTick

[C#]	protected	virtual	void	OnTick(EventArgs e);
[C++]	protected:	virtual	void	OnTick(EventArgs* e);
[VB]	Overridable	Protected	Sub	OnTick(ByVal e As EventArgs)
[JScript]	protected	function	OnTick(e :	EventArgs);

Description

Raises the **System.Windows.Forms.Timer.Tick** event.

This method is called for each timer tick. It calls any methods that are added through **System.Windows.Forms.Timer.Tick**. If you are inheriting from **System.Windows.Forms.Timer**, you can override this method. An **System.EventArgs** that contains the event data. This is always **System.EventArgs.Empty**.

q) Start

[C#]	public	void	Start();
[C++]	public:	void	Start();
[VB]	Public	Sub	Start()
[JScript]	public	function	Start();

Description

Starts the timer.

You can also start the timer by setting the **System.Windows.Forms.Timer.Enabled** property to **true** .

r) Stop

[C#]	public	void	Stop();
[C++]	public:	void	Stop();
[VB]	Public	Sub	Stop()
[JScript]	public	function	Stop();

Description

Stops the timer.

You can also stop the timer by setting the **System.Windows.Forms.Timer.Enabled** property to **false** . A timer that is disabled is subject to garbage collection.

s) ToString

[C#]	public	override	string	ToString();		
[C++]	public:	String*	ToString();			
[VB]	Overrides	Public	Function	ToString()	As	String
[JScript]	public	override	function	ToString()	:	String;

Description

returns us as a string.

ToolBar class (System.Windows.Forms)

a) *ToString*

Description

Represents a Windows toolbar.

System.Windows.Forms.ToolBar controls are used to display **System.Windows.Forms.ToolBarButton** controls that can appear as a standard button, a toggle-style button, or a drop-down style button. You can assign images to the buttons by instantiating an **System.Windows.Forms.ImageList** object, assigning it to the **System.Windows.Forms.ToolBar.ImageList** property of the toolbar, and assigning the image index value to the **System.Windows.Forms.ToolBarButton.ImageIndex** property of the button. You can then assign text to be displayed underneath or to the right of the image by setting the **System.Windows.Forms.ToolBarButton.Text** property of the **System.Windows.Forms.ToolBarButton**.

b) *ToolBar*

Example Syntax:

c) *ToString*

[C#]	public	ToolBar();
[C++]	public:	ToolBar();
[VB]	Public	Sub New()
[JScript]	public	function ToolBar();

Description

Initializes a new instance of the **System.Windows.Forms.ToolBar** class.

A newly instantiated toolbar control will be blank; add **System.Windows.Forms.ToolBarButton** controls by setting the **System.Windows.Forms.ToolBar.Buttons** property.

- d) *AccessibilityObject*
- e) *AccessibleDefaultActionDescription*
- f) *AccessibleDescription*
- g) *AccessibleName*
- h) *AccessibleRole*
- i) *AllowDrop*
- j) *Anchor*
- k) *Appearance*
- l) *ToString*

Description

Gets or set the value that determines the appearance of a toolbar control and its buttons.

The **System.Windows.Forms.ToolBar.Appearance** property affects the appearance of the buttons assigned to the toolbar. When the appearance is set to **ToolBarAppearance.Normal**, the toolbar's buttons appear three-dimensional and raised. Set the

System.Windows.Forms.ToolBar.Appearance property of the toolbar to **ToolBarAppearance.Flat** to give the toolbar's buttons a flat appearance. As the mouse pointer moves over the flat buttons, they appear raised and three-dimensional. Flat button separators appear as etched lines rather than spaces between the raised buttons. The flat style buttons will give your application a Web look.

m) *AutoSize*

n) *ToString*

[C#] public bool AutoSize {get; set;}

[C++] public: __property bool get_AutoSize();public: __property void
set_AutoSize(bool);

[VB] Public Property AutoSize As Boolean

[JScript] public function get AutoSize() : Boolean;public function set
AutoSize(Boolean);

Description

Gets or sets a value indicating whether the toolbar adjusts its size automatically, based on the size of the buttons and the dock style.

When **System.Windows.Forms.ToolBar.AutoSize** is set to **true** , the **System.Windows.Forms.ToolBar** control sizes itself to accommodate the toolbar buttons, based upon the button size and the **System.Windows.Forms.DockStyle** of the toolbar.

o) *BackColor*

p) *ToString*

[C#] public override Color BackColor {get; set;}

[C++] public: __property virtual Color get_BackColor();public: __property virtual
void set_BackColor(Color);

[VB] Overrides Public Property BackColor As Color

[JScript] public function get BackColor() : Color;public function set
BackColor(Color);

1
2 *Description*

3 *q) BackgroundImage*

4 *r) ToString*

5
6 [C#] public override Image BackgroundImage {get; set;}

7 [C++] public: __property virtual Image* get_BackgroundImage();public:

8 __property virtual void set_BackgroundImage(Image*);

9 [VB] Overrides Public Property BackgroundImage As Image

10 [JScript] public function get BackgroundImage() : Image;public function set

11 BackgroundImage(Image);

12
13 *Description*

14
15 *s) BindingContext*

16 *t) BorderStyle*

17 *u) ToString*

18
19
20 *Description*

21 Gets or sets the border style of the toolbar control.

22 The **System.Windows.Forms.ToolBar** can take on a sunken, three-
23 dimensional appearance when the **System.Windows.Forms.BorderStyle**
24 property is set to **BorderStyle.Fixed3D** . To display a flat thin border around
25 the toolbar control, set the **System.Windows.Forms.BorderStyle** property to
BorderStyle.FixedSingle .

- v) *Bottom*
- w) *Bounds*
- x) *Buttons*
- y) *ToString*

Description

Gets the collection of **System.Windows.Forms.ToolBarButton** controls assigned to the toolbar control.

The **System.Windows.Forms.ToolBar.Buttons** property is a zero-based indexed collection used to hold all the **System.Windows.Forms.ToolBarButton** controls assigned to the toolbar. Since the property is read-only, it can not be assigned a collection of toolbar buttons directly. Toolbar buttons may be added or removed by using the methods inherited from the **System.Windows.Forms.ToolBar.ToolBarButtonCollection** class. Use the **System.Windows.Forms.ToolBar.ToolBarButtonCollection.Add(System.Windows.Forms.ToolBarButton)** method to add individual buttons and the **System.Windows.Forms.ToolBar.ToolBarButtonCollection.Remove(System.Windows.Forms.ToolBarButton)** method to delete a button. Call the **System.Windows.Forms.ToolBar.ToolBarButtonCollection.Clear** method to remove all the buttons from the collection.

- z) *ButtonSize*
- aa) *ToString*

```
[C#]          public          Size          ButtonSize          {get;          set;}
[C++] public: __property Size get_ButtonSize();public: __property void
set_ButtonSize(Size);
[VB]          Public          Property          ButtonSize          As          Size
[JScript] public function get ButtonSize() : Size;public function set
```

1 ButtonSize(Size);

2
3 *Description*

4 Gets or sets the size of the buttons on the toolbar control.

5 If the **System.Windows.Forms.ToolBar.ButtonSize** is not set, it will be set
6 to its default, or alternatively, a **System.Drawing.Size** object is computed to
7 accommodate the largest **System.Drawing.Image** and text assigned to the
8 **System.Windows.Forms.ToolBarButton** controls.

9 *bb) CanFocus*

10 *cc) CanSelect*

11 *dd) Capture*

12 *ee) CausesValidation*

13 *ff) ClientRectangle*

14 *gg) ClientSize*

15 *hh) CompanyName*

16 *ii) Container*

17 *jj) ContainsFocus*

18 *kk) ContextMenu*

19 *ll) Controls*

20 *mm) Created*

21 *nn) CreateParams*

22 *oo) ToString*

23
24
25 *Description*

Returns the parameters needed to create the handle. Inheriting classes can override this to provide extra functionality. They should not, however, forget to get base.CreateParams first to get the struct filled up with the basic info.

pp) Cursor

qq) DataBindings

rr) DefaultImeMode

ss) ToString

Description

Gets the default Input Method Editor(IME) mode supported by this control.

As implemented in the **System.Windows.Forms.ToolBar** class, this property always returns the **System.Windows.Forms.ImeMode.Disable** value.

tt) DefaultSize

uu) ToString

[C#] protected override Size DefaultSize {get;}

[C++] protected: __property virtual Size get_DefaultSize();

[VB] Overrides Protected ReadOnly Property DefaultSize As Size

[JScript] protected function get DefaultSize() : Size;

Description

Deriving classes can override this to configure a default size for their control. This is more efficient than setting the size in the control's constructor.

vv) *DesignMode*

ww) *DisplayRectangle*

xx) *Disposing*

yy) *Divider*

zz) *ToString*

Description

Gets or sets a value indicating whether the toolbar displays a divider.

Dividers are displayed to help distinguish the toolbar from adjacent controls, such as menus.

aaa) *Dock*

bbb) *ToString*

```
[C#] public override DockStyle Dock {get; set;}
```

```
[C++] public: __property virtual DockStyle get_Dock();public: __property virtual  
void set_Dock(DockStyle);
```

```
[VB] Overrides Public Property Dock As DockStyle
```

```
[JScript] public function get Dock() : DockStyle;public function set  
Dock(DockStyle);
```

Description

Sets the way in which this ToolBar is docked to its parent. We need to override this to ensure autoSizing works correctly Sets the way in which this ToolBar is docked to its parent. We need to override this to ensure autoSizing works correctly

ccc) *DropDownArrows*

ddd) *ToString*

```
[C#]      public      bool      DropDownArrows      {get;      set;}
[C++] public: __property bool get_DropDownArrows();public: __property void
set_DropDownArrows(bool);
[VB]      Public      Property      DropDownArrows      As      Boolean
[JScript] public function get DropDownArrows() : Boolean;public function set
DropDownArrows(Boolean);
```

Description

Gets or sets a value indicating whether drop-down buttons on a toolbar display down arrows.

When **System.Windows.Forms.ToolBar.DropDownArrows** is set to **false** , no down arrows will display on drop-down style toolbar buttons. When the user clicks the drop-down button on the toolbar, the menu will drop down for selection. When the drop-down arrow is displayed, the user must press the down arrow to display the menu.

eee) *Enabled*

fff) *Events*

ggg) *Focused*

hhh) *Font*

iii) *FontHeight*

jjj) *ForeColor*

kkk) *ToString*

Description

lll) *Handle*

mmm) *HasChildren*

nnn) *Height*

ooo) *ImageList*

ppp) *ToString*

Description

Gets or sets the collection of images available to the toolbar button controls.

If you instantiate an **System.Windows.Forms.ImageList** object and assign it to the **System.Windows.Forms.ToolBar.ImageList** property, you will be able to assign an image from the list to the **System.Windows.Forms.ToolBarButton** controls by assigning the image's index value to the **System.Windows.Forms.ToolBarButton.ImageIndex** property of the toolbar button.

1 *qqq) ImageSize*

2 *rrr) ToString*

3
4 [C#] public Size ImageSize {get;}

5 [C++] public: __property Size get_ImageSize();

6 [VB] Public ReadOnly Property ImageSize As Size

7 [JScript] public function get ImageSize() : Size;

8
9 *Description*

10 Gets the size of the images in the image list assigned to the toolbar.

11 *sss) ImeMode*

12 *ttt) ToString*

13
14 [C#] public new ImeMode ImeMode {get; set;}

15 [C++] public: __property ImeMode get_ImeMode();public: __property void
16 set_ImeMode(ImeMode);

17 [VB] Public Property ImeMode As ImeMode

18 [JScript] public function get ImeMode() : ImeMode;public function set
19 ImeMode(ImeMode);

20
21 *Description*

22 Gets or sets the Input Method Editor(IME) mode supported by this control.

uuu) *InvokeRequired*

vvv) *IsAccessible*

www) *IsDisposed*

xxx) *IsHandleCreated*

yyy) *Left*

zzz) *Location*

aaaa) *Name*

bbbb) *Parent*

cccc) *ProductName*

dddd) *ProductVersion*

eeee) *RecreatingHandle*

ffff) *Region*

gggg) *RenderRightToLeft*

hhhh) *ResizeRedraw*

iiii) *Right*

jjjj) *RightToLeft*

kkkk) *ToString*

Description

1 *llll) ShowFocusCues*

2 *mmmm) ShowKeyboardCues*

3 *nnnn) ShowToolTips*

4 *oooo) ToString*

5
6
7 *Description*

8 Gets or sets a value indicating whether the toolbar displays a tool tip for each button.

9 To set the text displayed by the tool tip, set the
10 **System.Windows.Forms.ToolBarButton.ToolTipText** property of each
11 **System.Windows.Forms.ToolBarButton** on the
12 **System.Windows.Forms.ToolBar** . To cause the tool tip to display as the user
 moves the mouse pointer over the toolbar button, set the
13 **System.Windows.Forms.ToolBar.ShowToolTips** property to **true** .

13 *pppp) Site*

14 *qqqq) Size*

15 *rrrr) TabIndex*

16 *ssss) TabStop*

17 *tttt) ToString*

18
19
20
21 *Description*

22 **true** if XXX; otherwise, **false** . The default is XXX.

zzzz) *Top*

aaaaa) *TopLevelControl*

bbbbbb) *Visible*

ccccc) *Width*

dddddd) *WindowTarget*

eeeeee) *Wrappable*

fffff) *ToString*

Description

Gets or sets a value indicating whether the toolbar buttons wrap to the next line if the toolbar becomes too small to display all the buttons on the same line.

Toolbar buttons can be divided into logical groups by using separators. A separator is a toolbar button with the **System.Windows.Forms.ToolBarButton.Style** property set to **ToolBarButtonStyle.Separator** . If the **System.Windows.Forms.ToolBar.Wrappable** property is set to **true** and the toolbar becomes too small to display all the buttons on the same line, the toolbar will be broken into additional lines, with the breaks occurring at the separators. This ensures that button groups stay together. Toolbar buttons that are not in a group can be separated when the toolbar wraps. The toolbar may become too small to display all of its buttons on the same line if its parent **System.Windows.Forms.Form** is resized.

ggggg) *ToString*

Description

Occurs when a **System.Windows.Forms.ToolBarButton** on the **System.Windows.Forms.ToolBar** is clicked.

For more information about handling events, see .

hhhhh)ToString

```
[C#]   public   event   ToolBarButtonClickEventHandler   ButtonDropDown;
[C++]  public:  __event ToolBarButtonClickEventHandler*   ButtonDropDown;
[VB]   Public   Event   ButtonDropDown   As   ToolBarButtonClickEventHandler
```

Description

Occurs when a drop-down style **System.Windows.Forms.ToolBarButton** or its down arrow is clicked.

For more information about handling events, see .

iiii) CreateHandle

```
[C#]           protected           override           void           CreateHandle();
[C++]          protected:           void           CreateHandle();
[VB]           Overrides           Protected           Sub           CreateHandle()
[JScript]      protected           override           function           CreateHandle();
```

Description

jjjj) Dispose

```
[C#]           protected           override           void           Dispose(bool           disposing);
[C++]          protected:           void           Dispose(bool           disposing);
[VB]           Overrides           Protected           Sub           Dispose(ByVal           disposing   As   Boolean)
[JScript]      protected           override           function           Dispose(disposing   :   Boolean);
```


Description

kkkkk) OnButtonClick

[C#] protected virtual void OnButtonClick(ToolBarButtonClickEventArgs e);

[C++] protected: virtual void OnButtonClick(ToolBarButtonClickEventArgs* e);

[VB] Overridable Protected Sub OnButtonClick(ByVal e As ToolBarButtonClickEventArgs)

[JScript] protected function OnButtonClick(e : ToolBarButtonClickEventArgs);

Description

Raises the **System.Windows.Forms.ToolBar.ButtonClick** event.

Raising an event invokes the event handler through a delegate. For more information, see . A

System.Windows.Forms.ToolBarButtonClickEventArgs that contains the event data.

lllll) OnButtonDropDown

[C#] protected virtual void OnButtonDropDown(ToolBarButtonClickEventArgs e);

[C++] protected: virtual void OnButtonDropDown(ToolBarButtonClickEventArgs* e);

[VB] Overridable Protected Sub OnButtonDropDown(ByVal e As ToolBarButtonClickEventArgs)

[JScript] protected function OnButtonDropDown(e : ToolBarButtonClickEventArgs);

Description

Raises the **System.Windows.Forms.ToolBar.ButtonDropDown** event.

Raising an event invokes the event handler through a delegate. For more information, see . A

System.Windows.Forms.ToolBarButtonClickEventArgs that contains the event data.

mmmmm)OnFontChanged

[C#] protected override void OnFontChanged(EventArgs e);

[C++] protected: void OnFontChanged(EventArgs* e);

[VB] Overrides Protected Sub OnFontChanged(ByVal e As EventArgs)

[JScript] protected override function OnFontChanged(e : EventArgs);

Description

Overridden to ensure that the buttons and the control resize properly whenever the font changes. A

System.Windows.Forms.ToolBarButtonClickEventArgs that contains the event data.

nnnnn)OnHandleCreated

[C#] protected override void OnHandleCreated(EventArgs e);

[C++] protected: void OnHandleCreated(EventArgs* e);

[VB] Overrides Protected Sub OnHandleCreated(ByVal e As EventArgs)

[JScript] protected override function OnHandleCreated(e : EventArgs);

Description

Overridden from the control class so we can add all the buttons and do whatever work needs to be done. Don't forget to call `base.OnHandleCreated`. An **System.EventArgs** that contains the event data.

ooooo) OnResize

```
[C#]      protected      override      void      OnResize(EventArgs      e);
[C++]      protected:      void      OnResize(EventArgs*      e);
[VB]      Overrides      Protected      Sub      OnResize(ByVal e As EventArgs)
[JScript]      protected      override      function      OnResize(e : EventArgs);
```

Description

The control is being resized. Make sure the width/height are correct. An **System.EventArgs** that contains the event data.

ppppp) SetBoundsCore

```
[C#] protected override void SetBoundsCore(int x, int y, int width, int height,
BoundsSpecified specified);
[C++] protected: void SetBoundsCore(int x, int y, int width, int height,
BoundsSpecified specified);
[VB] Overrides Protected Sub SetBoundsCore(ByVal x As Integer, ByVal y As
Integer, ByVal width As Integer, ByVal height As Integer, ByVal specified As
BoundsSpecified)
[JScript] protected override function SetBoundsCore(x : int, y : int, width : int,
height : int, specified : BoundsSpecified);
```

Description

Overrides Control.setBoundsCore to enforce autoSize.

qqqqq) ToString

[C#] public override string ToString();
[C++] public: String* ToString();
[VB] Overrides Public Function ToString() As String
[JScript] public override function ToString() : String;

Description

Returns a string representation for this control.

Return Value: String Returns a string representation for this control.

rrrrr) WndProc

[C#] protected override void WndProc(ref Message m);
[C++] protected: void WndProc(Message* m);
[VB] Overrides Protected Sub WndProc(ByRef m As Message)
[JScript] protected override function WndProc(m : Message);

Description

ToolBarAppearance enumeration (System.Windows.Forms)

a) WndProc

Description

Specifies the type of toolbar to display.

This enumeration is used by members such as
System.Windows.Forms.ToolBar.Appearance .

b) WndProc

[C#]	public	const	ToolBarAppearance	Flat;
[C++]	public:	const	ToolBarAppearance	Flat;
[VB]	Public	Const	Flat As	ToolBarAppearance
[JScript]	public	var	Flat :	ToolBarAppearance;

Description

The toolbar and buttons appear flat, but the buttons change to three-dimensional as the mouse pointer moves over them.

c) WndProc

[C#]	public	const	ToolBarAppearance	Normal;
[C++]	public:	const	ToolBarAppearance	Normal;
[VB]	Public	Const	Normal As	ToolBarAppearance
[JScript]	public	var	Normal :	ToolBarAppearance;

Description

The toolbar and buttons appear as standard three-dimensional controls.

ToolBarButton class (System.Windows.Forms)

a) ToString

Description

Represents a Windows toolbar button.

System.Windows.Forms.ToolBarButton controls are parented by **System.Windows.Forms.ToolBar** controls. Common properties to set once the toolbar button has been instantiated are **System.Windows.Forms.ToolBarButton.Text** and **System.Windows.Forms.ToolBarButton.ImageIndex**. The text to be displayed underneath or to the right of the image can be set by setting the **System.Windows.Forms.ToolBarButton.Text** property of the button. You can assign images to the buttons by instantiating an **System.Windows.Forms.ImageList** object, assigning it to the **System.Windows.Forms.ToolBar.ImageList** property of the toolbar, and then assigning the image index value to the **System.Windows.Forms.ToolBarButton.ImageIndex** property of the button.

b) ToolBarButton

Example Syntax:

c) ToString

[C#]	public	ToolBarButton();
[C++]	public:	ToolBarButton();
[VB]	Public	Sub New()

[JScript] public function ToolBarButton(); Initializes a new instance of the **System.Windows.Forms.ToolBarButton** class.

Description

Initializes a new instance of the **System.Windows.Forms.ToolBarButton** class.

A newly instantiated **System.Windows.Forms.ToolBarButton** has no default **System.Windows.Forms.ToolBarButton.Text** or **System.Drawing.Image** assigned to it. The button's default style is **ToolBarButtonStyle.PushButton**.

d) *ToolBarButton*

Example Syntax:

e) *ToString*

```
[C#]          public          ToolBarButton(string          text);
[C++]          public:          ToolBarButton(String*          text);
[VB]      Public      Sub      New(ByVal      text      As      String)
[JScript]      public      function      ToolBarButton(text      :      String);
```

Description

Initializes a new instance of the **System.Windows.Forms.ToolBarButton** class and displays the assigned text on the button.

The newly instantiated **System.Windows.Forms.ToolBarButton** has no **System.Drawing.Image** assigned to it. The button's default style is **System.Windows.Forms.ToolBarButtonStyle.PushButton**. The *text* parameter is assigned to the **System.Windows.Forms.ToolBarButton.Text** property and is displayed on the new toolbar button control. The text to display on the new **System.Windows.Forms.ToolBarButton**.

f) *Container*

g) *DesignMode*

h) *DropDownMenu*

i) *ToString*

Description

Gets or sets the menu to be displayed in the drop-down toolbar button.

You can specify a **System.Windows.Forms.MenuItem** or context menu to be displayed when the drop-down button is pressed. This property is not used

unless the **System.Windows.Forms.ToolBarButtonStyle** is set to **DropDownButton** .

j) *Enabled*

k) *ToString*

[C#] public bool Enabled {get; set;}

[C++] public: __property bool get_Enabled();public: __property void
set_Enabled(bool);

[VB] Public Property Enabled As Boolean

[JScript] public function get Enabled() : Boolean;public function set
Enabled(Boolean);

Description

Gets or sets a value indicating whether the button is enabled.

When the **System.Windows.Forms.ToolBarButton.Enabled** property is set to **false** , the toolbar button cannot be pressed, and the button's appearance changes. The **System.Drawing.Image** and **System.Windows.Forms.ToolBarButton.Text** assigned to the button will appear grayed out. If the image or text has multiple colors, they will be displayed in a monochromatic gray.

l) *Events*

m) *ImageIndex*

n) *ToString*

Description

Gets or sets the index value of the image assigned to the button.

The **System.Windows.Forms.ToolBarButton.ImageIndex** value references the indexed value of the images in an **System.Windows.Forms.ImageList** assigned to the parent **System.Windows.Forms.ToolBar** control.

o) Parent

p) ToString

[C#]	public	ToolBar	Parent	{get;}
[C++]	public:	__property	ToolBar*	get_Parent();
[VB]	Public	ReadOnly	Property	Parent As ToolBar
[JScript]	public	function	get Parent()	: ToolBar;

Description

Gets the toolbar control that the toolbar button is assigned to.

q) PartialPush

r) ToString

[C#]	public	bool	PartialPush	{get; set;}
[C++]	public:	__property	bool	get_PartialPush();public: __property void set_PartialPush(bool);
[VB]	Public	Property	PartialPush	As Boolean
[JScript]	public	function	get PartialPush()	: Boolean;public function set PartialPush(Boolean);

Description

Gets or sets a value indicating whether a toggle-style toolbar button is partially pushed.

When **System.Windows.Forms.ToolBarButton.PartialPush** is set to **true** , the toolbar button appears to have its face grayed. This appearance is different from the dimmed appearance when the **System.Windows.Forms.ToolBarButton.Enabled** property is set to **false** since the partial-push appearance gives a haze to the entire button face. This property has no effect unless the **System.Windows.Forms.ToolBarButtonStyle** is set to **ToggleButton** .

s) *Pushed*

t) *ToString*

[C#] public bool Pushed {get; set;}

[C++] public: __property bool get_Pushed();public: __property void
set_Pushed(bool);

[VB] Public Property Pushed As Boolean

[JScript] public function get Pushed() : Boolean;public function set
Pushed(Boolean);

Description

Gets or sets a value indicating whether a toggle-style toolbar button is currently in the pushed state.

When **System.Windows.Forms.ToolBarButton.Pushed** is set to **true** , the toolbar button appears sunken or inset relative to the other buttons. This property has no effect unless the **System.Windows.Forms.ToolBarButtonStyle** is set to **ToggleButton** .

u) *Rectangle*

v) *ToString*

[C#] public Rectangle Rectangle {get;}

[C++] public: __property Rectangle get_Rectangle();

```

1  [VB]      Public      ReadOnly      Property      Rectangle      As      Rectangle
2  [JScript] public      function      get      Rectangle()      :      Rectangle;

```

Description

Gets the bounding rectangle for a toolbar button.

If the **System.Windows.Forms.ToolBar** and the current button are both **System.Windows.Forms.ToolBarButton.Visible** , then this property retrieves the bounding rectangle the button is currently contained in.

w) *Site*

x) *Style*

y) *ToString*

Description

Gets or sets the style of the toolbar button.

If the button **System.Windows.Forms.ToolBarButton.Style** is set to **System.Windows.Forms.ToolBarButtonStyle.DropDownButton** you can specify a **System.Windows.Forms.MenuItem** to be displayed when the drop-down button is pressed. If the style is set to **ToolBarButtonStyle.Separator** , the toolbar button appears as a button separator and not as a button. The **ToolBarButtonStyle.ToggleButton** style causes the toolbar button to act like a toggle button; it can be in an on or off state.

z) *Tag*

aa) *ToString*

```

23 [C#]      public      object      Tag      {get;      set;}
24 [C++]     public:      __property      Object*      get_Tag();public:      __property      void
25 set_Tag(Object*);

```

```

1 [VB]          Public          Property          Tag          As          Object
2 [JScript] public function get Tag() : Object;public function set Tag(Object);

```

Description

Gets or sets the object that contains data about the toolbar button.

Retrieves or assigns the data currently associated with the toolbar button. Any **System.Object** derived type may be assigned to this property. If this property is being set through the Windows Forms designer, only text may be assigned.

bb) Text

cc) ToString

```

11 [C#]          public          string          Text          {get;          set;}
12 [C++] public:  __property String* get_Text();public:  __property void
13 set_Text(String*);
14 [VB]          Public          Property          Text          As          String
15 [JScript] public function get Text() : String;public function set Text(String);

```

Description

Gets or sets the text displayed on the toolbar button.

The default the Text property value is an empty string ("") unless you created the control with the **System.Windows.Forms.ToolBar.#ctor** constructor that accepts the text string as a parameter. The orientation of the text on the toolbar button is determined by the **System.Windows.Forms.ToolBar.TextAlign** property of the button's parent **System.Windows.Forms.ToolBar**, which can be set to one of the **System.Windows.Forms.ToolBarTextAlign** enumeration values. The orientation is in relation to the image assigned to the button. If no image is assigned to the button, there will be space left for one on the surface of the toolbar button.

dd) *ToolTipText*

ee) *ToString*

[C#] public string ToolTipText {get; set;}

[C++] public: __property String* get_ToolTipText();public: __property void
set_ToolTipText(String*);

[VB] Public Property ToolTipText As String

[JScript] public function get ToolTipText() : String;public function set
ToolTipText(String);

Description

Gets or sets the text that appears as a tool tip for a control.

To enable the display of the tool tip text when the mouse pointer is moved over the button, set the **System.Windows.Forms.ToolBar.ShowToolTips** property of the button's parent **System.Windows.Forms.ToolBar** to **true** .

ff) *Visible*

gg) *ToString*

[C#] public bool Visible {get; set;}

[C++] public: __property bool get_Visible();public: __property void
set_Visible(bool);

[VB] Public Property Visible As Boolean

[JScript] public function get Visible() : Boolean;public function set
Visible(Boolean);

Description

Gets or sets a value indicating whether the toolbar button is visible.

If the toolbar button is not visible, it will not be displayed on the toolbar, and therefore cannot receive user input.

hh) Dispose

[C#] protected override void Dispose(bool disposing);

[C++] protected: void Dispose(bool disposing);

[VB] Overrides Protected Sub Dispose(ByVal disposing As Boolean)

[JScript] protected override function Dispose(disposing : Boolean);

Description

ii) ToString

[C#] public override string ToString();

[C++] public: String* ToString();

[VB] Overrides Public Function ToString() As String

[JScript] public override function ToString() : String;

Description

ToolBarButtonClickEventArgs class (System.Windows.Forms)

a) ToString

Description

Provides data for the **System.Windows.Forms.ToolBar.ButtonClick** event.

The event occurs whenever the user clicks on a button on a **System.Windows.Forms.ToolBar** control. The **System.Windows.Forms.ToolBarButtonClickEventArgs.button** property contains the **System.Windows.Forms.ToolBarButton** object with the information about the button that was clicked.

b) ToolBarButtonClickEventArgs

Example Syntax:

c) ToString

[C#] public ToolBarButtonClickEventArgs(ToolBarButton button);

[C++] public: ToolBarButtonClickEventArgs(ToolBarButton* button);

[VB] Public Sub New(ByVal button As ToolBarButton)

[JScript] public function ToolBarButtonClickEventArgs(button : ToolBarButton);

Description

Initializes a new instance of the **System.Windows.Forms.ToolBarButtonClickEventArgs** class.

The **System.Windows.Forms.ToolBarButtonClickEventArgs.button** property is set equal to the *button* parameter. The **System.Windows.Forms.ToolBarButton** that was clicked.

d) Button

e) ToString

[C#] public ToolBarButton Button {get; set;}

[C++] public: __property ToolBarButton* get_Button();public: __property void set_Button(ToolBarButton*);

[VB] Public Property Button As ToolBarButton

[JScript] public function get Button() : ToolBarButton;public function set

1 Button(ToolBarButton);

2
3 *Description*

4 Gets or sets the **System.Windows.Forms.ToolBarButton** that was clicked.

5 The **System.Windows.Forms.ToolBarButtonClickEventArgs.button**
6 property is initially set equal to the *button* parameter of the
System.Windows.Forms.ToolBarButtonClickEventArgs.#ctor constructor.

7 ToolBarButtonClickEventHandler delegate (System.Windows.Forms)

8 a) *ToString*

9
10
11 *Description*

12 Represents the method that will handle the
13 **System.Windows.Forms.ToolBar.ButtonClick** event of a
14 **System.Windows.Forms.ToolBar** . The source of the event. A
System.Windows.Forms.ToolBarButtonClickEventArgs that contains the
event data.

15 When you create a
16 **System.Windows.Forms.ToolBarButtonClickEventHandler** delegate, you
17 identify the method that will handle the event. To associate the event with your
18 event handler, add an instance of the delegate to the event. The event handler is
called whenever the event occurs, unless you remove the delegate. For more
information about handling events with delegates, see .

19 ToolBar.ToolBarButtonCollection class (System.Windows.Forms)

20 a) *ToString*

21
22
23 *Description*

24 Encapsulates a collection of **System.Windows.Forms.ToolBarButton** controls
25 for use by the **System.Windows.Forms.ToolBar** class.

The **System.Windows.Forms.ToolBar.ToolBarButtonCollection** is a zero-based indexed collection used by the **System.Windows.Forms.ToolBar** class to hold all the **System.Windows.Forms.ToolBarButton** controls assigned to the toolbar. Use the **System.Windows.Forms.ToolBar.ToolBarButtonCollection.Add(System.Windows.Forms.ToolBarButton)** method to add individual buttons and the **System.Windows.Forms.ToolBar.ToolBarButtonCollection.Remove(System.Windows.Forms.ToolBarButton)** method to delete them. Call the **System.Windows.Forms.ToolBar.ToolBarButtonCollection.Clear** method to remove all the buttons from the collection.

b) *ToolBar.ToolBarButtonCollection*

Example Syntax:

c) *ToString*

```
[C#]      public      ToolBar.ToolBarButtonCollection(ToolBar      owner);
[C++]      public:      ToolBarButtonCollection(ToolBar*      owner);
[VB]      Public      Sub      New(ByVal      owner      As      ToolBar)
[JScript] public function ToolBar.ToolBarButtonCollection(owner : ToolBar);
```

Description

Initializes a new instance of the **System.Windows.Forms.ToolBar.ToolBarButtonCollection** class and assigns it to the specified toolbar.

You do not need to instantiate the **System.Windows.Forms.ToolBar.ToolBarButtonCollection** object and explicitly call its constructor. When you assign the **System.Windows.Forms.ToolBar.Buttons** property of a **System.Windows.Forms.ToolBar** object, a **System.Windows.Forms.ToolBar.ToolBarButtonCollection** object will be created. You can then gain access to its properties and methods, and assign **System.Windows.Forms.ToolBarButton** controls to the collection. The **System.Windows.Forms.ToolBar** that parents the collection of **System.Windows.Forms.ToolBarButton** controls.

d) *Count*

e) *ToString*

```
[C#]          public          int          Count          {get;}
[C++]          public:          __property          int          get_Count();
[VB]    Public    ReadOnly    Property    Count    As    Integer
[JScript]    public    function    get    Count()    :    int;
```

Description

Gets the number of buttons in the toolbar button collection.

The **System.Windows.Forms.ToolBar.ToolBarButtonCollection.Count** property holds the actual count of the **System.Windows.Forms.ToolBarButton** controls assigned to the collection. It is common to use the **System.Windows.Forms.ToolBar.ToolBarButtonCollection.Count** property value as the upper bounds of a loop to iterate through a collection. Keep in mind that the index value of a collection is a zero-based index, so you must subtract one from the looping variable. If you do not account for this, you will exceed the upper bounds of the collection and throw an exception.

f) *IsReadOnly*

g) *ToString*

```
[C#]          public          bool          IsReadOnly          {get;}
[C++]          public:          __property          bool          get_IsReadOnly();
[VB]    Public    ReadOnly    Property    IsReadOnly    As    Boolean
[JScript]    public    function    get    IsReadOnly()    :    Boolean;
```

Description

Gets a value indicating whether the collection is read-only.

h) *Item*

i) *ToString*

[C#] public virtual ToolBarButton this[int index] {get; set;}

[C++] public: __property virtual ToolBarButton* get_Item(int index);public:

__property virtual void set_Item(int index, ToolBarButton*);

[VB] Overridable Public Default Property Item(ByVal index As Integer) As
ToolBarButton

[JScript] returnValue =

ToolBarButtonCollectionObject.Item(index);ToolBarButtonCollectionObject.Item
(index) = returnValue;

Description

Gets or sets the toolbar button at the specified indexed location in the toolbar
button collection.

To assign **System.Windows.Forms.ToolBarButton** controls to a specific
location, or to retrieve them from the
System.Windows.Forms.ToolBar.ToolBarButtonCollection , you can
reference the collection object with a specific index value. The index value of the
System.Windows.Forms.ToolBar.ToolBarButtonCollection is a zero-based
index. The indexed location of the **System.Windows.Forms.ToolBarButton**
in the collection.

j) *Add*

[C#] public int Add(string text);

[C++] public: int Add(String* text);

[VB] Public Function Add(ByVal text As String) As Integer

[JScript] public function Add(text : String) : int;

Description

Adds a new toolbar button to the end of the toolbar button collection with the specified **System.Windows.Forms.ToolBarButton.Text** property value.

Return Value: The zero-based index value of the **System.Windows.Forms.ToolBarButton** added to the collection.

You can also add new **System.Windows.Forms.ToolBarButton** objects to the collection by using the

System.Windows.Forms.ToolBar.ToolBarButtonCollection.AddRange(System.Windows.Forms.ToolBarButton[]) or

System.Windows.Forms.ToolBar.ToolBarButtonCollection.Insert(System.Int32, System.Windows.Forms.ToolBarButton) methods, or the other version of the

System.Windows.Forms.ToolBar.ToolBarButtonCollection.Add(System.Windows.Forms.ToolBarButton) method. The text to display on the new **System.Windows.Forms.ToolBarButton**.

k) Add

[C#] public int Add(ToolBarButton button);

[C++] public: int Add(ToolBarButton* button);

[VB] Public Function Add(ByVal button As ToolBarButton) As Integer

[JScript] public function Add(button : ToolBarButton) : int; Adds a new toolbar button to the end of the toolbar button collection.

Description

Adds a the specified toolbar button to the end of the toolbar button collection.

Return Value: The zero-based index value of the **System.Windows.Forms.ToolBarButton** added to the collection.

You can also add new **System.Windows.Forms.ToolBarButton** objects to the collection by using the

System.Windows.Forms.ToolBar.ToolBarButtonCollection.AddRange(System.Windows.Forms.ToolBarButton[]) or

System.Windows.Forms.ToolBar.ToolBarButtonCollection.Insert(System

m.Int32, System.Windows.Forms.ToolBarButton) methods, or the other version of the **System.Windows.Forms.ToolBar.ToolBarButtonCollection.Add(System.Windows.Forms.ToolBarButton)** method. The **System.Windows.Forms.ToolBarButton** to be added after all existing buttons.

l) *AddRange*

```
[C#]      public      void      AddRange(ToolBarButton[]      buttons);
[C++]     public:     void      AddRange(ToolBarButton*      buttons[]);
[VB]     Public  Sub  AddRange(ByVal  buttons()  As  ToolBarButton)
[JScript] public  function  AddRange(buttons  :  ToolBarButton[]);
```

Description

Adds a collection of toolbar buttons to this toolbar button collection.

The **System.Windows.Forms.ToolBarButton** objects contained in the *nodes* array are appended to the end of the collection. The collection of **System.Windows.Forms.ToolBarButton** controls to add to this **System.Windows.Forms.ToolBar.ToolBarButtonCollection** contained in an array.

m) *Clear*

```
[C#]      public      void      Clear();
[C++]     public:     __sealed      void      Clear();
[VB]     NotOverridable      Public      Sub      Clear()
[JScript] public      function      Clear();
```

Description

Removes all buttons from the toolbar button collection.

The **System.Windows.Forms.ToolBar.ToolBarButtonCollection.Clear** method iterates through the collection and removes all toolbar buttons assigned to the **System.Windows.Forms.ToolBar.ToolBarButtonCollection** .

n) Contains

[C#] public bool Contains(ToolBarButton button);

[C++] public: bool Contains(ToolBarButton* button);

[VB] Public Function Contains(ByVal button As ToolBarButton) As Boolean

[JScript] public function Contains(button : ToolBarButton) : Boolean;

Description

Determines if the specified toolbar button is a member of the collection.

Return Value: **true** if the **System.Windows.Forms.ToolBarButton** is a member of the collection; otherwise, **false** .

This method enables you to determine whether a **System.Windows.Forms.ToolBarButton** is member of the collection before attempting to perform operations on the **System.Windows.Forms.ToolBarButton** . You can use this method to confirm that a **System.Windows.Forms.ToolBarButton** has been added to or is still a member of the collection. The **System.Windows.Forms.ToolBarButton** to locate in the collection.

o) GetEnumerator

[C#] public IEnumerator GetEnumerator();

[C++] public: __sealed IEnumerator* GetEnumerator();

[VB] NotOverridable Public Function GetEnumerator() As IEnumerator

[JScript] public function GetEnumerator() : IEnumerator;

Description

Returns an enumerator that can be used to iterate through the toolbar button collection.

Return Value: An **System.Collections.IEnumerator** object that represents the tree node collection.

p) IndexOf

[C#] public int IndexOf(ToolBarButton button);

[C++] public: int IndexOf(ToolBarButton* button);

[VB] Public Function IndexOf(ByVal button As ToolBarButton) As Integer

[JScript] public function IndexOf(button : ToolBarButton) : int;

Description

Retrieves the index of the specified toolbar button in the collection.

Return Value: The zero-based index of the item found in the collection; otherwise, -1.

This method gives you easy access to the index value of the **System.Windows.Forms.ToolBarButton** in the collection. The index value allows you to easily determine which **System.Windows.Forms.ToolBarButton** was clicked on the **System.Windows.Forms.ToolBar**. The **System.Windows.Forms.ToolBarButton** clicked can be determined by evaluating the **System.Windows.Forms.ToolBar.ToolBarButtonCollection.IndexOf(System.Windows.Forms.ToolBarButton)** value of the **System.Windows.Forms.ToolBarButtonClickEventArgs.Button** property. The **System.Windows.Forms.ToolBarButton** to locate in the collection.

q) Insert

[C#] public void Insert(int index, ToolBarButton button);

[C++] public: void Insert(int index, ToolBarButton* button);

[VB] Public Sub Insert(ByVal index As Integer, ByVal button As ToolBarButton)

1 [JScript] public function Insert(index : int, button : ToolBarButton);

3 *Description*

4 Inserts an existing toolbar button in the toolbar button collection at the specified location.

5 You can also add new **System.Windows.Forms.ToolBarButton** objects to the collection by using the

6 **System.Windows.Forms.ToolBar.ToolBarButtonCollection.Add(System.Windows.Forms.ToolBarButton)** or
7 **System.Windows.Forms.ToolBar.ToolBarButtonCollection.AddRange(System.Windows.Forms.ToolBarButton[])** methods. The indexed location within the collection to insert the toolbar button. The
8 **System.Windows.Forms.ToolBarButton** to insert.

10 *r) Remove*

12 [C#] public void Remove(ToolBarButton button);

13 [C++] public: void Remove(ToolBarButton* button);

14 [VB] Public Sub Remove(ByVal button As ToolBarButton)

15 [JScript] public function Remove(button : ToolBarButton);

17 *Description*

18 Removes a given button from the toolbar button collection.

19 To remove toolbar buttons from the collection, use the

20 **System.Windows.Forms.ToolBar.ToolBarButtonCollection.RemoveAt(System.Int32)**, or

21 **System.Windows.Forms.ToolBar.ToolBarButtonCollection.Clear** methods. The **System.Windows.Forms.ToolBarButton** to remove from the collection.

s) *RemoveAt*

```
[C#]      public      void      RemoveAt(int      index);
[C++]     public:     __sealed   void      RemoveAt(int      index);
[VB]      NotOverridable Public Sub RemoveAt(ByVal index As Integer)
[JScript] public      function   RemoveAt(index      :      int);
```

Description

Removes a given button from the toolbar button collection.

The

System.Windows.Forms.ToolBar.ToolBarButtonCollection.Remove(System.Windows.Forms.ToolBarButton) method removes the **System.Windows.Forms.ToolBarButton** at the specified location in the **System.Windows.Forms.ToolBar.ToolBarButtonCollection**. If you wish to remove all **System.Windows.Forms.ToolBarButton** controls from the collection, use the **System.Windows.Forms.ToolBar.ToolBarButtonCollection.Clear** method. The indexed location of the **System.Windows.Forms.ToolBarButton** in the collection.

t) *ICollection.CopyTo*

```
[C#]      void      ICollection.CopyTo(Array      dest,      int      index);
[C++]     void      ICollection::CopyTo(Array*      dest,      int      index);
[VB]      Sub CopyTo(ByVal dest As Array, ByVal index As Integer) Implements
ICollection.CopyTo
[JScript] function ICollection.CopyTo(dest : Array, index : int);
```

u) *IList.Add*

```
[C#]      int      IList.Add(object      button);
```

1 [C++] int IList::Add(Object* button);
 2 [VB] Function Add(ByVal button As Object) As Integer Implements IList.Add
 3 [JScript] function IList.Add(button : Object) : int;

4 v) *IList.Contains*

5
 6 [C#] bool IList.Contains(object button);
 7 [C++] bool IList::Contains(Object* button);
 8 [VB] Function Contains(ByVal button As Object) As Boolean Implements
 9 IList.Contains
 10 [JScript] function IList.Contains(button : Object) : Boolean;

11 w) *IList.IndexOf*

12
 13 [C#] int IList.IndexOf(object button);
 14 [C++] int IList::IndexOf(Object* button);
 15 [VB] Function IndexOf(ByVal button As Object) As Integer Implements
 16 IList.IndexOf
 17 [JScript] function IList.IndexOf(button : Object) : int;

18 x) *IList.Insert*

19
 20 [C#] void IList.Insert(int index, object button);
 21 [C++] void IList::Insert(int index, Object* button);
 22 [VB] Sub Insert(ByVal index As Integer, ByVal button As Object) Implements
 23 IList.Insert
 24 [JScript] function IList.Insert(index : int, button : Object);
 25

y) *IList.Remove*

[C#] void IList.Remove(object button);
[C++] void IList::Remove(Object* button);
[VB] Sub Remove(ByVal button As Object) Implements IList.Remove
[JScript] function IList.Remove(button : Object);
ToolBarButtonStyle enumeration (System.Windows.Forms)

a) *ToString*

Description

Specifies the button style within a toolbar.

This enumeration is used by members such as
System.Windows.Forms.ToolBarButton.Style .

b) *ToString*

[C#] public const ToolBarButtonStyle DropDownButton;
[C++] public: const ToolBarButtonStyle DropDownButton;
[VB] Public Const DropDownButton As ToolBarButtonStyle
[JScript] public var DropDownButton : ToolBarButtonStyle;

Description

A drop-down control that displays a menu or other window when clicked.

c) *ToString*

```
[C#]      public      const      ToolBarButtonStyle      PushButton;
[C++]     public:     const      ToolBarButtonStyle      PushButton;
[VB]      Public      Const      PushButton      As      ToolBarButtonStyle
[JScript] public      var      PushButton      :      ToolBarButtonStyle;
```

Description

A standard, three-dimensional button.

d) *ToString*

```
[C#]      public      const      ToolBarButtonStyle      Separator;
[C++]     public:     const      ToolBarButtonStyle      Separator;
[VB]      Public      Const      Separator      As      ToolBarButtonStyle
[JScript] public      var      Separator      :      ToolBarButtonStyle;
```

Description

A space or line between toolbar buttons. The appearance depends on the value of the **System.Windows.Forms.ToolBar.Appearance** property.

e) *ToString*

```
[C#]      public      const      ToolBarButtonStyle      ToggleButton;
[C++]     public:     const      ToolBarButtonStyle      ToggleButton;
[VB]      Public      Const      ToggleButton      As      ToolBarButtonStyle
[JScript] public      var      ToggleButton      :      ToolBarButtonStyle;
```

1
2 *Description*

3 A toggle button that appears sunken when clicked and retains the sunken
4 appearance until clicked again.

5 ToolBarTextAlign enumeration (System.Windows.Forms)

6 *a) ToString*

7
8 *Description*

9 Specifies the alignment of text on the toolbar button control.

10 This enumeration is used by members such as
11 **System.Windows.Forms.ToolBar.TextAlign** .

12 *b) ToString*

13
14 [C#] public const ToolBarTextAlign Right;
15 [C++] public: const ToolBarTextAlign Right;
16 [VB] Public Const Right As ToolBarTextAlign
17 [JScript] public var Right : ToolBarTextAlign;
18

19 *Description*

20 The text is aligned to the right of the toolbar button image.

21 *c) ToString*

22
23 [C#] public const ToolBarTextAlign Underneath;
24 [C++] public: const ToolBarTextAlign Underneath;
25

[VB]	Public	Const	Underneath	As	ToolBarTextAlign
[JScript]	public	var	Underneath	:	ToolBarTextAlign;

Description

The text is aligned underneath the toolbar button image.

ToolTip class (System.Windows.Forms)

a) ToString

**System.Windows.Forms.ToolTipSystem.Windows.Forms.ToolTipSystem
 .Windows.Forms.TextBoxSystem.Windows.Forms.ToolTipSystem.Wind
 ows.Forms.ToolTipSystem.Windows.Forms.PictureBox**

b) ToolTip

Example Syntax:

c) ToString

Description

Initializes a new instance of the **System.Windows.Forms.ToolTip** class without a specified container.

This constructor associates the **System.Windows.Forms.ToolTip** class with the container that is creating the instance of the control.

d) ToolTip

Example Syntax:

e) ToString

[C#]	public	ToolTip(IContainer	cont);
[C++]	public:	ToolTip(IContainer*	cont);

```

1 [VB]      Public      Sub      New(ByVal      cont      As      IContainer)
2 [JScript] public function ToolTip(cont : IContainer); Initializes a new instance of
3 the      System.Windows.Forms.ToolTip      class.

```

Description

Initializes a new instance of the **System.Windows.Forms.ToolTip** class with a specified container.

This constructor enables you to associate a **System.Windows.Forms.ToolTip** with any container. An object that implements the **System.ComponentModel.IContainer** interface that represents the container of the **System.Windows.Forms.ToolTip** .

f) Active

g) ToString

```

13 [C#]      public      bool      Active      {get;      set;}

```

```

14 [C++]      public:      __property      bool      get_Active();public:      __property      void
15 set_Active(bool);

```

```

16 [VB]      Public      Property      Active      As      Boolean

```

```

17 [JScript] public function get Active() : Boolean;public function set
18 Active(Boolean);

```

Description

Gets or sets a value indicating whether the **System.Windows.Forms.ToolTip** is currently active.

This property allows you to enable or disable the display of ToolTip text for all controls that have text specified in this particular **System.Windows.Forms.ToolTip** . More than one **System.Windows.Forms.ToolTip** can be created and assigned to a form; setting the **System.Windows.Forms.ToolTip.Active** property to **false** only

affects the specified **System.Windows.Forms.ToolTip** . You can allow users to set the value of this property in a form that provides application options to provide the ability for the user to enable or disable the display of ToolTips in your application.

h) AutomaticDelay

i) ToString

```
[C#]      public      int      AutomaticDelay      {get;      set;}
```

```
[C++] public: __property int get_AutomaticDelay();public: __property void  
set_AutomaticDelay(int);
```

```
[VB]      Public      Property      AutomaticDelay      As      Integer
```

```
[JScript] public function get AutomaticDelay() : int;public function set  
AutomaticDelay(int);
```

Description

Gets or sets the automatic delay for the **System.Windows.Forms.ToolTip** .

The **System.Windows.Forms.ToolTip.AutomaticDelay** property enables you to set a single delay value which is then used to set the values of the **System.Windows.Forms.ToolTip.AutoPopDelay** , **System.Windows.Forms.ToolTip.InitialDelay** , and **System.Windows.Forms.ToolTip.ReshowDelay** properties. Each time the **System.Windows.Forms.ToolTip.AutomaticDelay** property is set, the following values are set by default.

j) AutoPopDelay

k) ToString

```
[C#]      public      int      AutoPopDelay      {get;      set;}
```

```
[C++] public: __property int get_AutoPopDelay();public: __property void  
set_AutoPopDelay(int);
```



```

1 [VB]      Public      Property      AutoPopDelay      As      Integer
2 [JScript] public function get AutoPopDelay() : int;public function set
3 AutoPopDelay(int);
4

```

Description

Gets or sets the period of time the **System.Windows.Forms.ToolTip** remains visible if the mouse pointer is stationary within a control with specified ToolTip text.

This property enables you to shorten or lengthen the time that the **System.Windows.Forms.ToolTip** window is displayed when the mouse pointer is over a control. For example, if you display extensive help in a ToolTip window, you can increase the value of this property to ensure that the user has sufficient time to read the text.

- l) *Container*
- m) *DesignMode*
- n) *Events*
- o) *InitialDelay*
- p) *ToString*

Description

Gets or sets the time that passes before the **System.Windows.Forms.ToolTip** appears.

This property enables you to shorten or lengthen the time that the **System.Windows.Forms.ToolTip** waits before displaying a ToolTip window. If the value of the **System.Windows.Forms.ToolTip.InitialDelay** property is set to a value that is too long in duration, the user of your application may not know that your application provides ToolTip help. You can use this property to ensure that the user has ToolTips displayed quickly by shortening the time specified.

q) *ReshowDelay*

r) *ToString*

[C#] public int ReshowDelay {get; set;}

[C++] public: __property int get_ReshowDelay();public: __property void
set_ReshowDelay(int);

[VB] Public Property ReshowDelay As Integer

[JScript] public function get ReshowDelay() : int;public function set
ReshowDelay(int);

Description

Gets or sets the length of time that must transpire before subsequent ToolTip windows appear as the mouse pointer moves from one control to another.

This property enables you to shorten or lengthen the time that the **System.Windows.Forms.ToolTip** waits before displaying a ToolTip window after a previous ToolTip window is displayed. The first time a ToolTip window is displayed the value of the **System.Windows.Forms.ToolTip.InitialDelay** property is used to determine the delay to apply before initially showing the ToolTip window. When a ToolTip window is currently being displayed and the user moves the cursor to another control that displays a ToolTip window, the value of the **System.Windows.Forms.ToolTip.ReshowDelay** property is used before showing the ToolTip for the new control. The ToolTip window from the previous control must still be displayed in order for the delay specified in the **System.Windows.Forms.ToolTip.ReshowDelay** property to be used, otherwise the **System.Windows.Forms.ToolTip.InitialDelay** property value is used.

s) *ShowAlways*

t) *ToString*

[C#] public bool ShowAlways {get; set;}

[C++] public: __property bool get_ShowAlways();public: __property void

1 set_ShowAlways(bool);

2 [VB] Public Property ShowAlways As Boolean

3 [JScript] public function get ShowAlways() : Boolean; public function set

4 ShowAlways(Boolean);

5
6 *Description*

7 Gets or sets a value indicating whether the **System.Windows.Forms.ToolTip**
appears even when its parent control is not active.

8 This property enables you to display a ToolTip window even when the container
9 of the **System.Windows.Forms.ToolTip** is not active. You can use this feature
10 in a modeless window application to enable ToolTip windows to be displayed
11 regardless of which modeless window is active. This feature is also useful when
12 creating a control using the **System.Windows.Forms.UserControl** that
13 contains a number of controls within it that display ToolTip windows. Since the
System.Windows.Forms.UserControl is often not the active window on a
form, setting this property to **true** enables the controls within the
System.Windows.Forms.UserControl to display ToolTip windows at any
time.

14 u) *Site*

15 v) *CanExtend*

16
17 [C#] public bool CanExtend(object target);

18 [C++] public: __sealed bool CanExtend(Object* target);

19 [VB] NotOverridable Public Function CanExtend(ByVal target As Object) As
20 Boolean

21 [JScript] public function CanExtend(target : Object) : Boolean;

22
23 *Description*

24 Returns true if the ToolTip can offer an extender property to the specified target
25 component.

Return Value: True if this ToolTip can offer one or more extender properties.
Target object to add extender property to.

w) *Dispose*

```
[C#]      protected      override      void      Dispose(bool      disposing);
[C++]      protected:      void      Dispose(bool      disposing);
[VB] Overrides Protected Sub Dispose(ByVal disposing As Boolean)
[JScript] protected override function Dispose(disposing : Boolean);
```

Description

Disposes of the **System.Windows.Forms.ToolTip** component.

If the **System.Windows.Forms.ToolTip** component has a site, this method removes the component from its container and raises the Dispose event.

x) *Finalize*

```
[C#] ~ToolTip();
[C++] ~ToolTip();
[VB] Overrides Protected Sub Finalize()
[JScript] protected override function Finalize(); Finalizes garbage collection.
```

y) *GetToolTip*

```
[C#]      public      string      GetToolTip(Control      control);
[C++]      public:      String*      GetToolTip(Control*      control);
[VB] Public Function GetToolTip(ByVal control As Control) As String
[JScript] public function GetToolTip(control : Control) : String;
```

Description

Retrieves the **System.Windows.Forms.ToolTip** text associated with the specified control.

Return Value: The **System.Windows.Forms.ToolTip** text for the specified control.

This method enables you to retrieve the ToolTip text for any control. If the ToolTip text changes dramatically in an application, you can use this method to find out what text is displayed at any point within the application when a control displays different text depending on the state of the application. To change the text that a control is displaying, use the **System.Windows.Forms.ToolTip.SetToolTip(System.Windows.Forms.Control, System.String)** method. The control for which to retrieve the **System.Windows.Forms.ToolTip** text.

z) RemoveAll

[C#]	public	void	RemoveAll();
[C++]	public:	void	RemoveAll();
[VB]	Public	Sub	RemoveAll()
[JScript]	public	function	RemoveAll();

Description

Removes all ToolTip text currently associated with the **System.Windows.Forms.ToolTip** control.

You can use this method to remove all ToolTip text that is associated with the **System.Windows.Forms.ToolTip** control. To disable the display of text instead of removing all ToolTip text from the **System.Windows.Forms.ToolTip** control, use the **System.Windows.Forms.ToolTip.Active** property.

aa) *SetToolTip*

```
[C#]    public    void    SetToolTip(Control    control,    string    caption);  
[C++]    public:    void    SetToolTip(Control*    control,    String*    caption);  
[VB]    Public Sub SetToolTip(ByVal control As Control, ByVal caption As String)  
[JScript]    public    function    SetToolTip(control : Control, caption : String);
```

Description

Associates **System.Windows.Forms.ToolTip** text with the specified control.

In addition to setting the ToolTip text to display for a control, you can also use this method to change the text to display for a control. Calling the **System.Windows.Forms.ToolTip.SetToolTip(System.Windows.Forms.Control, System.String)** method more than once for a given control does not specify multiple ToolTip text to display for a control but instead changes the current ToolTip text for the control. To determine the ToolTip text that is associated with a control at runtime, you can use the **System.Windows.Forms.ToolTip.GetToolTip(System.Windows.Forms.Control)** method. The **System.Windows.Forms.Control** to associate the ToolTip text with. The ToolTip text to display when the mouse cursor is over the control.

bb) *ToString*

```
[C#]          public          override          string          ToString();  
[C++]          public:          String*          ToString();  
[VB]    Overrides    Public    Function    ToString()    As    String  
[JScript]    public    override    function    ToString()    :    String;
```

Description

Returns a string representation for this control.

Return Value: String Returns a string representation for this control.

1 **TrackBar** class (System.Windows.Forms)

2 *a) ToString*

3
4
5 *Description*

6 Represents a standard Windows track bar.

7 The **System.Windows.Forms.TrackBar** is a scrollable control similar to the
8 **System.Windows.Forms.ScrollBar** control. You can configure ranges through
9 which the value of the **System.Windows.Forms.TrackBar.Value** property of
10 a track bar scrolls by setting the
11 **System.Windows.Forms.TrackBar.Minimum** property to specify the lower
12 end of the the range and the **System.Windows.Forms.TrackBar.Maximum**
13 property to specify the upper end of the range.

11 *b) TrackBar*

12 *Example Syntax:*

13 *c) ToString*

15 [C#]	public	TrackBar();
16 [C++]	public:	TrackBar();
17 [VB]	Public	Sub New()
18 [JScript]	public	function TrackBar();

19
20 *Description*

21 Initializes a new instance of the **System.Windows.Forms.TrackBar** class.

22 The track bar is created with a default horizontal orientation and a range of zero
23 to 10, with a tick mark shown for every value.

- d) *AccessibilityObject*
- e) *AccessibleDefaultActionDescription*
- f) *AccessibleDescription*
- g) *AccessibleName*
- h) *AccessibleRole*
- i) *AllowDrop*
- j) *Anchor*
- k) *AutoSize*
- l) *ToString*

Description

Gets or sets a value indicating whether the control's height or width is being automatically sized.

You can set the **System.Windows.Forms.TrackBar.AutoSize** property to **true** to cause the track bar to adjust either its height or width, depending on orientation, to ensure that the control uses only the required amount of space.

- m) *BackColor*
- n) *BackgroundImage*
- o) *ToString*

Description

- p) *BindingContext*
- q) *Bottom*
- r) *Bounds*
- s) *CanFocus*
- t) *CanSelect*
- u) *Capture*
- v) *CausesValidation*
- w) *ClientRectangle*
- x) *ClientSize*
- y) *CompanyName*
- z) *Container*
- aa) *ContainsFocus*
- bb) *ContextMenu*
- cc) *Controls*
- dd) *Created*
- ee) *CreateParams*
- ff) *ToString*

Description

This is called when creating a window. Inheriting classes can override this to add extra functionality, but should not forget to first call `base.CreateParams()` to make sure the control continues to work correctly.

Return Value: a `CreateParams` object with the information for creating this Control.

1 *gg) Cursor*

2 *hh) DataBindings*

3 *ii) DefaultImeMode*

4 *jj) ToString*

5
6
7 *Description*

8 Gets a value indicating the mode for the Input Method Editor (IME) for the
9 **System.Windows.Forms.TrackBar** .

10 *kk) DefaultSize*

11 *ll) ToString*

12 [C#] protected override Size DefaultSize {get;}
13

14 [C++] protected: __property virtual Size get_DefaultSize();
15

16 [VB] Overrides Protected ReadOnly Property DefaultSize As Size
17

18 [JScript] protected function get DefaultSize() : Size;
19

20
21
22 *Description*

23 Deriving classes can override this to configure a default size for their control.
24 This is more efficient than setting the size in the control's constructor.
25

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25

- mm) DesignMode*
- nn) DisplayRectangle*
- oo) Disposing*
- pp) Dock*
- qq) Enabled*
- rr) Events*
- ss) Focused*
- tt) Font*
- uu) ToString*

Description

- vv) FontHeight*
- ww) ForeColor*
- xx) ToString*

Description

Gets the foreground color of the track bar.

yy) *Handle*

zz) *HasChildren*

aaa) *Height*

bbb) *ImeMode*

ccc) *ToString*

Description

Gets or sets the Input Method Editor (IME) mode supported by this control.

ddd) *InvokeRequired*

eee) *IsAccessible*

fff) *IsDisposed*

ggg) *IsHandleCreated*

hhh) *LargeChange*

iii) *ToString*

Description

Gets or sets a value to be added to or subtracted from the **System.Windows.Forms.TrackBar.Value** property when the scroll box is moved a large distance.

When the user presses the PAGE UP or PAGE DOWN key or clicks the track bar on either side of the slider, the **System.Windows.Forms.TrackBar.Value** property changes according to the value set in the **System.Windows.Forms.TrackBar.LargeChange** property. You might consider setting the **System.Windows.Forms.TrackBar.LargeChange** value to a percentage of the **System.Windows.Forms.Control.Height** (for a vertically oriented track bar) or **System.Windows.Forms.Control.Width** (for

a horizontally oriented track bar) values. This keeps the distance your track bar moves in proportion to its size.

jjj) Left

kkk) Location

lll) Maximum

mmm) ToString

Description

Gets or sets the upper limit of the range this **System.Windows.Forms.TrackBar** is working with.

You can use the

System.Windows.Forms.TrackBar.SetRange(System.Int32, System.Int32) method to set both the **System.Windows.Forms.TrackBar.Maximum** and **System.Windows.Forms.TrackBar.Minimum** properties at the same time.

nnn) Minimum

ooo) ToString

[C#] public int Minimum {get; set;}

[C++] public: __property int get_Minimum();public: __property void
set_Minimum(int);

[VB] Public Property Minimum As Integer

[JScript] public function get Minimum() : int;public function set Minimum(int);

Description

Gets or sets the lower limit of the range this **System.Windows.Forms.TrackBar** is working with.

You can use the

System.Windows.Forms.TrackBar.SetRange(System.Int32, System.Int32) method to set both the **System.Windows.Forms.TrackBar.Maximum** and **System.Windows.Forms.TrackBar.Minimum** properties at the same time.

ppp) Name

qqq) Orientation

rrr) ToString

Description

Gets or sets a value indicating the horizontal or vertical orientation of the track bar.

When the **System.Windows.Forms.TrackBar.Orientation** property is set to **System.Windows.Forms.Orientation.Horizontal**, the slider moves from right to left as the **System.Windows.Forms.TrackBar.Value** increases. When the **System.Windows.Forms.TrackBar.Orientation** property is set to **System.Windows.Forms.Orientation.Vertical**, the slider moves from bottom to top as the **System.Windows.Forms.TrackBar.Value** increases.

1 *sss) Parent*
 2 *ttt) ProductName*
 3 *uuu) ProductVersion*
 4 *vvv) RecreatingHandle*
 5 *www) Region*
 6 *xxx) RenderRightToLeft*
 7 *yyy) ResizeRedraw*
 8 *zzz) Right*
 9 *aaaa) RightToLeft*
 10 *bbbb) ShowFocusCues*
 11 *cccc) ShowKeyboardCues*
 12 *dddd) Site*
 13 *eeee) Size*
 14 *ffff) SmallChange*
 15 *gggg) ToString*

17
18
19 *Description*

20 Gets or sets the value added to or subtracted from to the
 21 **System.Windows.Forms.TrackBar.Value** property when the slider is moved
 a small distance.

22 When the user presses one of the arrow keys, the
 23 **System.Windows.Forms.TrackBar.Value** property changes according to the
 value set in the **System.Windows.Forms.TrackBar.SmallChange** property.

1 *hhhh) TabIndex*

2 *iiii) TabStop*

3 *jjjj) Tag*

4 *kkkk) Text*

5 *llll) ToString*

6
7
8 *Description*

9
10 *mmmm)TickFrequency*

11 *nnnn) ToString*

12
13 [C#] public int TickFrequency {get; set;}

14 [C++] public: __property int get_TickFrequency();public: __property void
15 set_TickFrequency(int);

16 [VB] Public Property TickFrequency As Integer

17 [JScript] public function get TickFrequency() : int;public function set
18 TickFrequency(int);

19
20 *Description*

21 Gets or sets a value that specifyies the delta between ticks drawn on the control.

22 For a **System.Windows.Forms.TrackBar** with a large range of values
23 between the **System.Windows.Forms.TrackBar.Minimum** and the
24 **System.Windows.Forms.TrackBar.Maximum** , it might be impractical to
25 draw all the ticks for values on the control. For example, if you have a control
 with a range of 100, passing in a value of five here causes the control to draw 20
 ticks. In this case each tick would represent 5 units in the range of values.

oooo) *TickStyle*

pppp) *ToString*

```
[C#]      public      TickStyle      TickStyle      {get;      set;}
[C++] public: __property TickStyle get_TickStyle();public: __property void
set_TickStyle(TickStyle);
[VB]      Public      Property      TickStyle      As      TickStyle
[JScript] public function get TickStyle() : TickStyle;public function set
TickStyle(TickStyle);
```

Description

Gets or sets a value indicating how to display the tick marks on the track bar.

You can use the **System.Windows.Forms.TrackBar.TickStyle** property to modify the manner in which the tick marks are displayed on the track bar.

qqqq) *Top*

rrrr) *TopLevelControl*

ssss) *Value*

tttt) *ToString*

Description

Gets or sets a numeric value that represents the current position of the slider on the track bar control.

The **System.Windows.Forms.TrackBar.Value** property contains the number that represents the current position of the slider on the track bar.

uuuu) *Visible*

vvvv) *Width*

www) *WindowTarget*

xxxx) *ToString*

yyyy) *ToString*

zzzz) *ToString*

Description

Occurs when either a mouse or keyboard action moves the slider.

When you create a **System.Windows.Forms.TrackBar.Scroll** delegate, you identify the method that will handle the event. To associate the event with your event handler, add an instance of the delegate to the event. The event handler is called whenever the event occurs, unless you remove the delegate.

aaaaa) *ToString*

Description

Occurs when the **System.Windows.Forms.TrackBar.Value** property of a track bar changes, either by movement of the slider or by manipulation in code.

You can use this event to update other controls when the value represented in the track bar changes.

bbbb) *BeginInit*

[C#]	public	void	BeginInit();	
[C++]	public:	__sealed	void	BeginInit();
[VB]	NotOverridable	Public	Sub	BeginInit()

[JScript]	public	function	BeginInit();
-----------	--------	----------	--------------

Description

Begins the initialization of a **System.Windows.Forms.TrackBar** that is used on a form or used by another component. The initialization occurs at run time.

The Visual Studio.NET design environment uses this method to start the initialization of a component that is used on a form or used by another component. The **System.Windows.Forms.TrackBar.EndInit** method ends the initialization. Using **System.Windows.Forms.TrackBar.BeginInit** and **System.Windows.Forms.TrackBar.EndInit** methods prevents the control from being used before it is fully initialized.

ccccc) CreateHandle

[C#]	protected	override	void	CreateHandle();
[C++]	protected:		void	CreateHandle();
[VB]	Overrides	Protected	Sub	CreateHandle()
[JScript]	protected	override	function	CreateHandle();

Description

dddddd) EndInit

[C#]	public		void	EndInit();
[C++]	public:	__sealed	void	EndInit();
[VB]	NotOverridable	Public	Sub	EndInit()
[JScript]	public		function	EndInit();

Description

Ends the initialization of a **System.Windows.Forms.TrackBar** that is used on a form or used by another component. The initialization occurs at run time.

The Visual Studio.NET design environment uses this method to end the initialization of a component that is used on a form or used by another component. The **System.Windows.Forms.TrackBar.BeginInit** method starts the initialization. Using the **System.Windows.Forms.TrackBar.BeginInit** and **System.Windows.Forms.TrackBar.EndInit** methods prevents the control from being used before it is fully initialized.

eeee) IsInputKey

[C#] protected override bool IsInputKey(Keys keyData);

[C++] protected: bool IsInputKey(Keys keyData);

[VB] Overrides Protected Function IsInputKey(ByVal keyData As Keys) As Boolean

[JScript] protected override function IsInputKey(keyData : Keys) : Boolean;

Description

Handling special input keys, such as pgup, pgdown, home, end, etc...

ffff) OnBackColorChanged

[C#] protected override void OnBackColorChanged(EventArgs e);

[C++] protected: void OnBackColorChanged(EventArgs* e);

[VB] Overrides Protected Sub OnBackColorChanged(ByVal e As EventArgs)

[JScript] protected override function OnBackColorChanged(e : EventArgs);

Description

This method is called by the control when any property changes. Inheriting controls can override this method to get property change notification on basic properties. Inheriting controls must call base.propertyChanged.

ggggg) OnHandleCreated

```
[C#]    protected    override    void    OnHandleCreated(EventArgs    e);
[C++]    protected:    void    OnHandleCreated(EventArgs*    e);
[VB]    Overrides Protected Sub OnHandleCreated(ByVal e As EventArgs)
[JScript] protected override function OnHandleCreated(e : EventArgs);
```

Description

hhhhh) OnScroll

```
[C#]    protected    virtual    void    OnScroll(EventArgs    e);
[C++]    protected:    virtual    void    OnScroll(EventArgs*    e);
[VB]    Overridable Protected Sub OnScroll(ByVal e As EventArgs)
[JScript]    protected    function    OnScroll(e : EventArgs);
```

Description

Raises the **Scroll** event.

You can use the

System.Windows.Forms.TrackBar.OnScroll(System.EventArgs) event to update other controls as the position of the slider changes. An **System.EventArgs** that contains the event data.

iiii) OnValueChanged

```
[C#]    protected    virtual    void    OnValueChanged(EventArgs    e);
[C++]    protected:    virtual    void    OnValueChanged(EventArgs*    e);
[VB]    Overridable Protected Sub OnValueChanged(ByVal e As EventArgs)
```

1 [JScript] protected function OnValueChanged(e : EventArgs);

2
3 *Description*

4 Actually fires the "valueChanged" event.

5 *jjjjj) SetBoundsCore*

6
7 [C#] protected override void SetBoundsCore(int x, int y, int width, int height,
8 BoundsSpecified specified);

9 [C++] protected: void SetBoundsCore(int x, int y, int width, int height,
10 BoundsSpecified specified);

11 [VB] Overrides Protected Sub SetBoundsCore(ByVal x As Integer, ByVal y As
12 Integer, ByVal width As Integer, ByVal height As Integer, ByVal specified As
13 BoundsSpecified)

14 [JScript] protected override function SetBoundsCore(x : int, y : int, width : int,
15 height : int, specified : BoundsSpecified);

16
17 *Description*

18 Overrides Control.setBoundsCore to enforce autoSize.

19 *kkkkk) SetRange*

20
21 [C#] public void SetRange(int minValue, int maxValue);

22 [C++] public: void SetRange(int minValue, int maxValue);

23 [VB] Public Sub SetRange(ByVal minValue As Integer, ByVal maxValue As
24 Integer)

25 [JScript] public function SetRange(minValue : int, maxValue : int);

Description

Sets the minimum and maximum values for a **System.Windows.Forms.TrackBar** control.

You can use this method to set the entire range for the **System.Windows.Forms.TrackBar** control at the same time. To set the minimum or maximum values individually use the **System.Windows.Forms.TrackBar.Minimum** and **System.Windows.Forms.TrackBar.Maximum** properties. If the *minValue* parameter is greater than the *maxValue* parameter, the *maxValue* is set equal to *minValue*. The lower limit of the range of the track bar. The upper limit of the range of the track bar.

IIII) ToString

[C#]	public	override	string	ToString();
[C++]	public:		String*	ToString();
[VB]	Overrides	Public	Function	ToString() As String
[JScript]	public	override	function	ToString() : String;

Description

Returns a string representation for this control.

Return Value: String Returns a string representation for this control.

mmmmm)WndProc

[C#]	protected	override	void	WndProc(ref Message m);
[C++]	protected:		void	WndProc(Message* m);
[VB]	Overrides	Protected	Sub	WndProc(ByRef m As Message)
[JScript]	protected	override	function	WndProc(m : Message);

Description

The button's window procedure. Inheriting classes can override this to add extra functionality, but should not forget to call `base.wndProc(m)`; to ensure the button continues to function properly. A Windows Message Object.

TreeNode class (System.Windows.Forms)

a) WndProc

Description

Represents a node of a **System.Windows.Forms.TreeView** .

The **System.Windows.Forms.TreeNode.Nodes** collection holds all the child **System.Windows.Forms.TreeNode** objects assigned to the current **System.Windows.Forms.TreeNode** . You can add, remove, or clone a **System.Windows.Forms.TreeNode** ; when doing so, all child tree nodes are added, removed, or cloned. Each **System.Windows.Forms.TreeNode** can contain a collection of other **System.Windows.Forms.TreeNode** objects. This can make it difficult to determine where you are in the **System.Windows.Forms.TreeView** when iterating through the collection. To determine your location in a tree structure, use the **System.Windows.Forms.TreeNode.FullPath** property. The **System.Windows.Forms.TreeNode.FullPath** string can be parsed using the **System.Windows.Forms.TreeView.PathSeparator** string value to determine where a **System.Windows.Forms.TreeNode** label begins and ends.

b) TreeNode

Example Syntax:

c) WndProc

[C#]	public	TreeNode();
[C++]	public:	TreeNode();

g) *WndProc*

```
[C#]    public    TreeNode(string    text,    TreeNode[]    children);  
[C++]   public:   TreeNode(String*    text,    TreeNode*    children[]);  
[VB]    Public Sub New(ByVal text As String, ByVal children() As TreeNode)  
[JScript] public function TreeNode(text : String, children : TreeNode[]);
```

Description

Initializes a new instance of the **System.Windows.Forms.TreeNode** class with the specified label text and child tree nodes.

The *text* parameter value is assigned to the node's **System.Windows.Forms.TreeNode.Text** property and becomes the tree node label. The tree nodes contained in the *children* array are added to the **System.Windows.Forms.TreeNodeCollection** stored in the **System.Windows.Forms.TreeNode.Nodes** property. The label **System.Windows.Forms.TreeNode.Text** of the new tree node. An array of child **System.Windows.Forms.TreeNode** objects.

h) *TreeNode*

Example Syntax:

i) *WndProc*

```
[C#]    public    TreeNode(string    text,    int    imageIndex,    int    selectedIndex);  
[C++]   public:   TreeNode(String*    text,    int    imageIndex,    int    selectedIndex);  
[VB]    Public Sub New(ByVal text As String, ByVal imageIndex As Integer, ByVal  
selectedImageIndex As Integer)  
[JScript] public function TreeNode(text : String, imageIndex : int,  
selectedImageIndex : int);
```

Description

Initializes a new instance of the **System.Windows.Forms.TreeNode** class with the specified label text and images to display when the tree node is in a selected and unselected state.

The *text* parameter value is assigned to the node's **System.Windows.Forms.TreeNode.Text** property and becomes the tree node label. The *imageIndex* and *selectedImageIndex* values are the index values of an **System.Drawing.Image** stored in the **System.Windows.Forms.ImageList** assigned to the **System.Windows.Forms.TreeView.ImageList** property. The image referenced in the *imageIndex* property is displayed when the tree node is not selected. Likewise, the image referenced in the *selectedImageIndex* property is displayed when the tree node is in a selected state. The label **System.Windows.Forms.TreeNode.Text** of the new tree node. The index value of **System.Drawing.Image** to display when the tree node is unselected. The index value of **System.Drawing.Image** to display when the tree node is selected.

j) *TreeNode*

Example Syntax:

k) *WndProc*

```
[C#] public TreeNode(string text, int imageIndex, int selectedImageIndex,  
TreeNode[] children);
```

```
[C++] public: TreeNode(String* text, int imageIndex, int selectedImageIndex,  
TreeNode* children[]);
```

```
[VB] Public Sub New(ByVal text As String, ByVal imageIndex As Integer, ByVal  
selectedImageIndex As Integer, ByVal children() As TreeNode)
```

```
[JScript] public function TreeNode(text : String, imageIndex : int,  
selectedImageIndex : int, children : TreeNode[]);
```

Description

Initializes a new instance of the **System.Windows.Forms.TreeNode** class with the specified label text, child tree nodes, and images to display when the tree node is in a selected and unselected state.

The *text* parameter value is assigned to the node's **System.Windows.Forms.TreeNode.Text** property and becomes the tree node label. The *imageIndex* and *selectedImageIndex* values are the index values of an **System.Drawing.Image** stored in the **System.Windows.Forms.ImageList** assigned to the **System.Windows.Forms.TreeView.ImageList** property. The image referenced in the *imageIndex* parameter is displayed when the tree node is not selected. Likewise, the image referenced in the *selectedImageIndex* parameter is displayed when the tree node is in a selected state. The tree nodes contained in the *children* array are added to the **System.Windows.Forms.TreeNodeCollection** stored in the **System.Windows.Forms.TreeNode.Nodes** property. The label **System.Windows.Forms.TreeNode.Text** of the new tree node. The index value of **System.Drawing.Image** to display when the tree node is unselected. The index value of **System.Drawing.Image** to display when the tree node is selected. An array of child **System.Windows.Forms.TreeNode** objects.

l) *BackColor*

m) *WndProc*

```
[C#]          public          Color          BackColor          {get;          set;}
```

```
[C++] public: __property Color get_BackColor();public: __property void  
set_BackColor(Color);
```

```
[VB]          Public          Property          BackColor          As          Color
```

```
[JScript] public function get BackColor() : Color;public function set  
BackColor(Color);
```

Description

Gets or sets the background color of the tree node.

If the **System.Windows.Forms.TreeNode.BackColor** property is set to **null** , the **System.Drawing.Color** used is the **System.Windows.Forms.Control.BackColor** property value of the **System.Windows.Forms.TreeView** control that the tree node is assigned to.

n) Bounds

o) WndProc

```
[C#]          public          Rectangle          Bounds          {get;}
```

```
[C++]          public:          __property          Rectangle          get_Bounds();
```

```
[VB]          Public          ReadOnly          Property          Bounds          As          Rectangle
```

```
[JScript]          public          function          get          Bounds()          :          Rectangle;
```

Description

Gets the bounds of the tree node.

The coordinates are relative to the upper left corner of the **System.Windows.Forms.TreeView** control.

p) Checked

q) WndProc

```
[C#]          public          bool          Checked          {get;          set;}
```

```
[C++]          public:          __property          bool          get_Checked();public:          __property          void          set_Checked(bool);
```

```
[VB]          Public          Property          Checked          As          Boolean
```

```
[JScript]          public          function          get          Checked()          :          Boolean;public          function          set          Checked(Boolean);
```

Description

Gets or sets a value indicating whether the tree node is in a checked state.

r) *FirstNode*

s) *WndProc*

[C#] public TreeNode FirstNode {get;}

[C++] public: __property TreeNode* get_FirstNode();

[VB] Public ReadOnly Property FirstNode As TreeNode

[JScript] public function get FirstNode() : TreeNode;

Description

Gets the first child tree node in the tree node collection.

The **System.Windows.Forms.TreeNode.FirstNode** is the first child **System.Windows.Forms.TreeNode** in the **System.Windows.Forms.TreeNodeCollection** stored in the **System.Windows.Forms.TreeNode.Nodes** property of the current tree node. If the **System.Windows.Forms.TreeNode** has no child tree node, the **System.Windows.Forms.TreeNode.FirstNode** property returns **null**.

t) *ForeColor*

u) *WndProc*

[C#] public Color ForeColor {get; set;}

[C++] public: __property Color get_ForeColor();public: __property void set_ForeColor(Color);

[VB] Public Property ForeColor As Color

[JScript] public function get ForeColor() : Color;public function set

1 ForeColor(Color);

3 *Description*

4 Gets or sets the foreground color of the tree node.

5 If **null** , the **System.Drawing.Color** used is the
6 **System.Windows.Forms.Control.ForeColor** property value of the
System.Windows.Forms.TreeView control that the tree node is assigned to.

7 v) *FullPath*

8 w) *WndProc*

10 [C#] public string FullPath {get;}

11 [C++] public: __property String* get_FullPath();

12 [VB] Public ReadOnly Property FullPath As String

13 [JScript] public function get FullPath() : String;

15 *Description*

16 Gets the path from the root tree node to the current tree node.

17 The path consists of the labels of all of the tree nodes that must be navigated to
18 get to this tree node, starting at the root tree node. The node labels are
separated by the delimiter character specified in the

19 **System.Windows.Forms.TreeView.PathSeparator** property of the
System.Windows.Forms.TreeView control that contains this node. For
20 example, if the delimiter character of the tree view control named "Location" is
set to the backslash character, (\), the

21 **System.Windows.Forms.TreeNode.FullPath** property value is
"Country\Region\State".

x) *Handle*

y) *WndProc*

[C#] public IntPtr Handle {get;}

[C++] public: __property IntPtr get_Handle();

[VB] Public ReadOnly Property Handle As IntPtr

[JScript] public function get Handle() : IntPtr;

Description

Gets the handle of the tree node.

If a handle is not already created when the **System.Windows.Forms.TreeNode.Handle** property is referenced, it is created.

z) *ImageIndex*

aa) *WndProc*

[C#] public int ImageIndex {get; set;}

[C++] public: __property int get_ImageIndex();public: __property void
set_ImageIndex(int);

[VB] Public Property ImageIndex As Integer

[JScript] public function get ImageIndex() : int;public function set
ImageIndex(int);

Description

Gets or sets the image list index value of the image displayed when the tree node is in the unselected state.

The **System.Windows.Forms.TreeNode.ImageIndex** value is the index value of an **System.Drawing.Image** stored in the **System.Windows.Forms.ImageList** assigned to the **System.Windows.Forms.TreeView.ImageList** property.

bb) Index

cc) WndProc

[C#]	public	int	Index	{get;}
[C++]	public:	__property	int	get_Index();
[VB]	Public	ReadOnly	Property	Index As Integer
[JScript]	public	function	get	Index() : int;

Description

Gets the position of the tree node in the tree node collection.

dd) IsEditing

ee) WndProc

[C#]	public	bool	IsEditing	{get;}
[C++]	public:	__property	bool	get_IsEditing();
[VB]	Public	ReadOnly	Property	IsEditing As Boolean
[JScript]	public	function	get	IsEditing() : Boolean;

Description

Gets a value indicating whether the tree node is in an editable state.

ff) IsExpanded

gg) WndProc

```
[C#]          public          bool          IsExpanded          {get;}
[C++]         public:         __property      bool          get_IsExpanded();
[VB]   Public   ReadOnly   Property   IsExpanded   As   Boolean
[JScript]     public   function   get   IsExpanded()   :   Boolean;
```

Description

Gets a value indicating whether the tree node is in the expanded state.

hh) IsSelected

ii) WndProc

```
[C#]          public          bool          IsSelected          {get;}
[C++]         public:         __property      bool          get_IsSelected();
[VB]   Public   ReadOnly   Property   IsSelected   As   Boolean
[JScript]     public   function   get   IsSelected()   :   Boolean;
```

Description

Gets a value indicating whether the tree node is in the selected state.

jj) IsVisible

kk) WndProc

```
[C#]          public          bool          IsVisible          {get;}
[C++]         public:         __property      bool          get_IsVisible();
```

```

1  [VB]      Public      ReadOnly      Property      IsVisible      As      Boolean
2  [JScript]      public      function      get      IsVisible()      :      Boolean;

```

Description

Gets a value indicating whether the tree node is visible.

ll) LastNode

mm) WndProc

```

9  [C#]      public      TreeNode      LastNode      {get;}
10 [C++]      public:      __property      TreeNode*      get_LastNode();
11 [VB]      Public      ReadOnly      Property      LastNode      As      TreeNode
12 [JScript]      public      function      get      LastNode()      :      TreeNode;

```

Description

Gets the last child tree node.

The **System.Windows.Forms.TreeNode.LastNode** is the last child **System.Windows.Forms.TreeNode** in the **System.Windows.Forms.TreeNodeCollection** stored in the **System.Windows.Forms.TreeNode.Nodes** property of the current tree node. If the **System.Windows.Forms.TreeNode** has no child tree node, the **System.Windows.Forms.TreeNode.LastNode** property returns **null**.

nn) NextNode

oo) WndProc

```

23 [C#]      public      TreeNode      NextNode      {get;}
24 [C++]      public:      __property      TreeNode*      get_NextNode();
25 [VB]      Public      ReadOnly      Property      NextNode      As      TreeNode

```

1 [JScript] public function get NextNode() : TreeNode;

3 *Description*

4 Gets the next sibling tree node.

5 The **System.Windows.Forms.TreeNode.NextNode** is the next sibling
6 **System.Windows.Forms.TreeNode** in the
7 **System.Windows.Forms.TreeNodeCollection** stored in the
8 **System.Windows.Forms.TreeNode.Nodes** property of the tree node's parent
9 **System.Windows.Forms.TreeNode** . If there is no next tree node, the
10 **System.Windows.Forms.TreeNode.NextNode** property returns **null** .

11 *pp) NextVisibleNode*

12 *qq) WndProc*

13 [C#] public TreeNode NextVisibleNode {get;}

14 [C++] public: __property TreeNode* get_NextVisibleNode();

15 [VB] Public ReadOnly Property NextVisibleNode As TreeNode

16 [JScript] public function get NextVisibleNode() : TreeNode;

17 *Description*

18 Gets the next visible tree node.

19 The **System.Windows.Forms.TreeNode.NextVisibleNode** can be a child,
20 sibling, or a tree node from another branch. If there is no next tree node, the
21 **System.Windows.Forms.TreeNode.NextVisibleNode** property returns **null**
22 .

23 *rr) NodeFont*

24 *ss) WndProc*

25 [C#] public Font NodeFont {get; set;}

1 [C++] public: __property Font* get_NodeFont();public: __property void
2 set_NodeFont(Font*);

3 [VB] Public Property NodeFont As Font

4 [JScript] public function get NodeFont() : Font;public function set
5 NodeFont(Font);

6
7 *Description*

8 Gets or sets the font used to display the text on the tree node's label.

9 If **null** , the **System.Drawing.Font** used is the
10 **System.Windows.Forms.Control.Font** property value of the
System.Windows.Forms.TreeView control that this node is attached to.

11 *tt) Nodes*

12 *uu) WndProc*

13
14 [C#] public TreeNodeCollection Nodes {get;}

15 [C++] public: __property TreeNodeCollection* get_Nodes();

16 [VB] Public ReadOnly Property Nodes As TreeNodeCollection

17 [JScript] public function get Nodes() : TreeNodeCollection;

18
19 *Description*

20 Gets the collection of **System.Windows.Forms.TreeNode** objects assigned to
21 the current tree node.

22 The **System.Windows.Forms.TreeNode.Nodes** property can hold a collection
23 of other **System.Windows.Forms.TreeNode** objects. Each of the tree node in
24 the collection has a **System.Windows.Forms.TreeNode.Nodes** property that
25 can contain its own **System.Windows.Forms.TreeNodeCollection** . This
nesting of tree nodes can make it difficult to navigate a tree structure. The
System.Windows.Forms.TreeNode.FullPath property makes it easier to
determine your location in a tree.

vv) *Parent*

ww) *WndProc*

[C#] public TreeNode Parent {get;}

[C++] public: __property TreeNode* get_Parent();

[VB] Public ReadOnly Property Parent As TreeNode

[JScript] public function get Parent() : TreeNode;

Description

Gets the parent tree node of the current tree node.

If the tree node is at the root level, the

System.Windows.Forms.TreeNode.Parent property returns **null** .

xx) *PrevNode*

yy) *WndProc*

[C#] public TreeNode PrevNode {get;}

[C++] public: __property TreeNode* get_PrevNode();

[VB] Public ReadOnly Property PrevNode As TreeNode

[JScript] public function get PrevNode() : TreeNode;

Description

Gets the previous sibling tree node.

The **System.Windows.Forms.TreeNode.PrevNode** is the previous sibling

System.Windows.Forms.TreeNode in the

System.Windows.Forms.TreeNodeCollection stored in the

System.Windows.Forms.TreeNode.Nodes property of the tree node's parent

System.Windows.Forms.TreeNode . If there is no previous tree node, the

System.Windows.Forms.TreeNode.PrevNode property returns **null** .

zz) *PrevVisibleNode*

aaa) *WndProc*

```
[C#]      public      TreeNode      PrevVisibleNode      {get;}
[C++]     public:      __property      TreeNode*      get_PrevVisibleNode();
[VB]      Public      ReadOnly      Property      PrevVisibleNode      As      TreeNode
[JavaScript] public      function      get      PrevVisibleNode()      :      TreeNode;
```

Description

Gets the previous visible tree node.

The **System.Windows.Forms.TreeNode.PrevVisibleNode** can be a child, sibling, or a tree node from another branch. If there is no previous tree node, the **System.Windows.Forms.TreeNode.PrevVisibleNode** property returns **null**.

bbb) *SelectedImageIndex*

ccc) *WndProc*

```
[C#]      public      int      SelectedImageIndex      {get;      set;}
[C++]     public:      __property      int      get_SelectedImageIndex();public:      __property      void
set_SelectedImageIndex(int);
[VB]      Public      Property      SelectedImageIndex      As      Integer
[JavaScript] public      function      get      SelectedImageIndex()      :      int;public      function      set
SelectedImageIndex(int);
```

Description

Gets or sets the image list index value of the image that is displayed when the tree node is in the selected state.

The **System.Windows.Forms.TreeNode.SelectedIndex** value is the index value of an **System.Drawing.Image** stored in the **System.Windows.Forms.ImageList** assigned to the **System.Windows.Forms.TreeView.ImageList** property.

ddd) Tag

eee) WndProc

[C#] public object Tag {get; set;}

[C++] public: __property Object* get_Tag();public: __property void
set_Tag(Object*);

[VB] Public Property Tag As Object

[JScript] public function get Tag() : Object;public function set Tag(Object);

Description

Gets or sets the object that contains data about the tree node.

Any **System.Object** derived type may be assigned to this property. If this property is being set through the Windows Forms designer, only text may be assigned.

fff) Text

ggg) WndProc

[C#] public string Text {get; set;}

[C++] public: __property String* get_Text();public: __property void
set_Text(String*);

[VB] Public Property Text As String

[JScript] public function get Text() : String;public function set Text(String);

Description

Gets or sets the text displayed in the label of the tree node.

This property cannot be set by the user if the **System.Windows.Forms.TreeView.LabelEdit** property of the parent **System.Windows.Forms.TreeView** is set to **false** . The alternative to setting this property explicitly is to create the tree node using one of the **System.Windows.Forms.TreeNode.#ctor** constructors that has a string parameter that represents the **System.Windows.Forms.TreeNode.Text** property. The label is displayed next to the **System.Windows.Forms.TreeNode** image, if one is displayed.

hhh) TreeView

iii) WndProc

[C#]	public	TreeView	TreeView	{get;}
[C++]	public:	__property	TreeView*	get_TreeView();
[VB]	Public	ReadOnly	Property	TreeView As TreeView
[JScript]	public	function	get	TreeView() : TreeView;

Description

Gets the parent tree view that the tree node is assigned to.

jjj) BeginEdit

[C#]	public	void	BeginEdit();
[C++]	public:	void	BeginEdit();
[VB]	Public	Sub	BeginEdit()
[JScript]	public	function	BeginEdit();

Description

Initiates the editing of the tree node label.

A common place to use this method is to call it on the **System.Windows.Forms.MenuItem.Click** event of a **System.Windows.Forms.MenuItem** or **System.Windows.Forms.ContextMenu**.

kkk) Clone

[C#]	public	virtual	object	Clone();
[C++]	public:	virtual	Object*	Clone();
[VB]	Overridable	Public	Function	Clone() As Object
[JScript]	public	function	Clone()	: Object;

Description

Copies the tree node and the entire subtree rooted at this tree node.

Return Value: The **System.Object** that represents the cloned **System.Windows.Forms.TreeNode** object.

The tree structure from the tree node being cloned and below is copied. Any child tree nodes assigned to the **System.Windows.Forms.TreeNode** being cloned is included in the new tree node and subtree.

lll) Collapse

[C#]	public	void	Collapse();
[C++]	public:	void	Collapse();
[VB]	Public	Sub	Collapse()
[JScript]	public	function	Collapse();

Description

Collapses the tree node.

The **System.Windows.Forms.TreeNode.Collapse** method only collapses the current **System.Windows.Forms.TreeNode** ; child tree nodes are not collapsed.

mmm) EndEdit

[C#] public void EndEdit(bool cancel);

[C++] public: void EndEdit(bool cancel);

[VB] Public Sub EndEdit(ByVal cancel As Boolean)

[JScript] public function EndEdit(cancel : Boolean);

Description

Ends the editing of the tree node label. **true** if the editing of the tree node label text was canceled without being saved; otherwise, **false**.

nnn) EnsureVisible

[C#] public void EnsureVisible();

[C++] public: void EnsureVisible();

[VB] Public Sub EnsureVisible()

[JScript] public function EnsureVisible();

Description

Ensures that the tree node is visible, expanding tree nodes and scrolling the tree view control as necessary.

When the **System.Windows.Forms.TreeNode.EnsureVisible** method is called, the tree is expanded and scrolled to ensure that the current tree node is visible in the **System.Windows.Forms.TreeView** . This method is useful if you are selecting a tree node in code based on certain criteria. By calling this method after selecting the node, it ensures that the user can see and interact with the selected node.

ooo) Expand

[C#]	public	void	Expand();
[C++]	public:	void	Expand();
[VB]	Public	Sub	Expand()
[JScript]	public	function	. Expand();

Description

Expands the tree node.

The **System.Windows.Forms.TreeNode.Expand** method expands the current **System.Windows.Forms.TreeNode** down to the next level of nodes.

ppp) ExpandAll

[C#]	public	void	ExpandAll();
[C++]	public:	void	ExpandAll();
[VB]	Public	Sub	ExpandAll()
[JScript]	public	function	ExpandAll();

Description

Expands all the child tree nodes.

The **System.Windows.Forms.TreeNode.ExpandAll** method expands all the child tree nodes assigned to the **System.Windows.Forms.TreeNode.Nodes** collection.

qqq) *FromHandle*

```
1
2 [C#] public static TreeNode FromHandle(TreeView tree, IntPtr handle);
3 [C++] public: static TreeNode* FromHandle(TreeView* tree, IntPtr handle);
4 [VB] Public Shared Function FromHandle(ByVal tree As TreeView, ByVal
5 handle As IntPtr) As TreeNode
6 [JScript] public static function FromHandle(tree : TreeView, handle : IntPtr) :
7 TreeNode;
```

9 *Description*

10 Returns the tree node with the specified handle and assigned to the specified
11 tree view control.

12 *Return Value:* A **System.Windows.Forms.TreeNode** that represents the tree
13 node assigned to the specified **System.Windows.Forms.TreeView** control
with the specified handle. The **System.Windows.Forms.TreeView** that
contains the tree node. The handle of the tree node.

14 rrr) *GetNodeCount*

```
15
16 [C#] public int GetNodeCount(bool includeSubTrees);
17 [C++] public: int GetNodeCount(bool includeSubTrees);
18 [VB] Public Function GetNodeCount(ByVal includeSubTrees As Boolean) As
19 Integer
20 [JScript] public function GetNodeCount(includeSubTrees : Boolean) : int;
```

22 *Description*

23 Returns the number of child tree nodes.

24 *Return Value:* The number of child tree nodes assigned to the
25 **System.Windows.Forms.TreeNode.Nodes** collection. **true** if the resulting

count includes all tree nodes indirectly rooted at this tree node; otherwise, **false**

1

.

2

sss) Remove

3

4

[C#]	public	void	Remove();
------	--------	------	-----------

5

[C++]	public:	void	Remove();
-------	---------	------	-----------

6

[VB]	Public	Sub	Remove()
------	--------	-----	----------

7

[JScript]	public	function	Remove();
-----------	--------	----------	-----------

8

9

Description

10

Removes the current tree node from the tree view control.

11

When the **System.Windows.Forms.TreeNode.Remove** method is called, the tree node and any child tree nodes assigned to the

12

System.Windows.Forms.TreeNode are removed from the

13

System.Windows.Forms.TreeView . The removed child nodes are removed from the **System.Windows.Forms.TreeView** , but are still attached to this tree node.

14

15

ttt) Toggle

16

17

[C#]	public	void	Toggle();
------	--------	------	-----------

18

[C++]	public:	void	Toggle();
-------	---------	------	-----------

19

[VB]	Public	Sub	Toggle()
------	--------	-----	----------

20

[JScript]	public	function	Toggle();
-----------	--------	----------	-----------

21

22

Description

23

Toggles the tree node to either the expanded or collapsed state.

24

The tree node is toggled to the state opposite its current state, either expanded or collapsed.

25

uuu) ToString

[C#]	public	override	string	ToString();
[C++]	public:		String*	ToString();
[VB]	Overrides	Public	Function	ToString() As String
[JScript]	public	override	function	ToString() : String;

Description

Returns the label text for the tree node Returns the label text for the tree node

TreeNodeCollection class (System.Windows.Forms)

a) ToString

Description

Represents a collection of **System.Windows.Forms.TreeNode** objects.

The **System.Windows.Forms.TreeNodeCollection.Add(System.String)** , **System.Windows.Forms.TreeNodeCollection.Remove(System.Windows.Forms.TreeNode)** , and

System.Windows.Forms.TreeNodeCollection.RemoveAt(System.Int32) methods enable you to add and remove individual tree nodes from the collection. You can also use the

System.Windows.Forms.TreeNodeCollection.AddRange(System.Windows.Forms.TreeNode[]) or

System.Windows.Forms.TreeNodeCollection.Clear methods to add or remove all the tree nodes from the collection.

b) Count

c) ToString

[C#]	public	int	Count	{get;}
------	--------	-----	-------	--------

```

1  [C++]      public:      __property      int      get_Count();
2  [VB]      Public      ReadOnly      Property      Count      As      Integer
3  [JScript]      public      function      get      Count()      :      int;

```

Description

Gets the total number of **System.Windows.Forms.TreeNode** objects in the collection.

The **System.Windows.Forms.TreeNodeCollection.Count** property holds the number of **System.Windows.Forms.TreeNode** objects assigned to the collection. You can use the

System.Windows.Forms.TreeNodeCollection.Count property value as the upper bounds of a loop to iterate through a collection.

d) IsReadOnly

e) ToString

```

14 [C#]      public      bool      IsReadOnly      {get;}
15 [C++]      public:      __property      bool      get_IsReadOnly();
16 [VB]      Public      ReadOnly      Property      IsReadOnly      As      Boolean
17 [JScript]      public      function      get      IsReadOnly()      :      Boolean;

```

Description

Gets a value indicating whether the collection is read-only.

f) Item

g) ToString

```

24 [C#]      public      virtual      TreeNode      this[int      index]      {get;      set;}
25 [C++]      public:      __property      virtual      TreeNode*      get_Item(int      index);public:

```



```

1  __property      virtual      void      set_Item(int      index,      TreeNode*);
2  [VB] Overridable Public Default Property Item(ByVal index As Integer) As
3  TreeNode
4  [JScript]                                returnValue      =
5  TreeNodeCollectionObject.Item(index);TreeNodeCollectionObject.Item(index) =
6  returnValue;

```

Description

Indicates the **System.Windows.Forms.TreeNode** at the specified indexed location in the collection.

To assign **System.Windows.Forms.TreeNode** objects to a specific location, or to retrieve them from the **System.Windows.Forms.TreeNodeCollection**, you can reference the collection object with a specific index value. The index value of the **System.Windows.Forms.TreeNodeCollection** is a zero-based index. The indexed location of the **System.Windows.Forms.TreeNode** in the collection.

h) Add

```

16 [C#]      public      virtual      TreeNode      Add(string      text);
17 [C++]     public:     virtual      TreeNode*      Add(String*      text);
18 [VB] Overridable Public Function Add(ByVal text As String) As TreeNode
19 [JScript] public function Add(text : String) : TreeNode; Adds a new tree node to
20 the                                             collection.

```

Description

Adds a new tree node to the end of the current tree node collection with the specified label text.

Return Value: A **System.Windows.Forms.TreeNode** that represents the tree node being added to the collection.

You can also add new **System.Windows.Forms.TreeNode** objects to the collection by using the **System.Windows.Forms.TreeNodeCollection.AddRange(System.Windows.Forms.TreeNode[])** or **System.Windows.Forms.TreeNodeCollection.Insert(System.Int32, System.Windows.Forms.TreeNode)** methods. The label text displayed by the **System.Windows.Forms.TreeNode** .

i) Add

```
[C#]      public      virtual      int      Add(TreeNode      node);
[C++]     public:     virtual      int      Add(TreeNode*      node);
[VB]      Overridable Public Function Add(ByVal node As TreeNode) As Integer
[JScript] public      function      Add(node      :      TreeNode)      :      int;
```

Description

Adds a previously created tree node to the end of the tree node collection.

Return Value: The zero-based index value of the

System.Windows.Forms.TreeNode added to the tree node collection.

This version of the

System.Windows.Forms.TreeNodeCollection.Add(System.String)

method allows you to add previously created

System.Windows.Forms.TreeNode objects to the end of the tree node

collection. The **System.Windows.Forms.TreeNode** to add to the collection.

j) AddRange

```
[C#]      public      virtual      void      AddRange(TreeNode[]      nodes);
[C++]     public:     virtual      void      AddRange(TreeNode*      nodes[]);
[VB]      Overridable Public Sub AddRange(ByVal nodes() As TreeNode)
[JScript] public      function      AddRange(nodes      :      TreeNode[]);
```

Description

Adds an array of previously created tree nodes to the collection.

The **System.Windows.Forms.TreeNode** objects contained in the *nodes* array are appended to the end of the collection. An array of **System.Windows.Forms.TreeNode** objects representing the tree nodes to add to the collection.

k) Clear

[C#]	public	virtual	void	Clear();
[C++]	public:	virtual	void	Clear();
[VB]	Overridable	Public	Sub	Clear()
[JScript]	public	function		Clear();

Description

Removes all tree nodes from the collection.

You can use this method to clear the entire collection of tree nodes from a tree view.

l) Contains

[C#]	public	bool	Contains(TreeNode	node);
[C++]	public:	bool	Contains(TreeNode*	node);
[VB]	Public Function	Contains(ByVal node As TreeNode)	As Boolean	
[JScript]	public	function	Contains(node : TreeNode)	: Boolean;

Description

Determines whether the specified tree node is a member of the collection.

Return Value: **true** if the **System.Windows.Forms.TreeNode** is a member of the collection; otherwise, **false** .

This method enables you to determine whether a

System.Windows.Forms.TreeNode is a member of the collection before attempting to perform operations on the **System.Windows.Forms.TreeNode** . You can use this method to confirm that a

System.Windows.Forms.TreeNode has been added to or is still a member of the collection. The **System.Windows.Forms.TreeNode** to locate in the collection.

m) CopyTo

[C#] public void CopyTo(Array dest, int index);

[C++] public: __sealed void CopyTo(Array* dest, int index);

[VB] NotOverridable Public Sub CopyTo(ByVal dest As Array, ByVal index As Integer)

[JScript] public function CopyTo(dest : Array, index : int);

Description

Copies the entire collection into an existing array at a specified location within the array. The destination array. The index in the destination array at which storing begins.

n) GetEnumerator

[C#] public IEnumerator GetEnumerator();

[C++] public: __sealed IEnumerator* GetEnumerator();

[VB] NotOverridable Public Function GetEnumerator() As IEnumerator

[JScript] public function GetEnumerator() : IEnumerator;

1
2 *Description*

3 Returns an enumerator that can be used to iterate through the tree node
4 collection.

5 *Return Value:* An **System.Collections.IEnumerator** object that represents the
6 tree node collection.

7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
o) *IndexOf*

[C#] public int IndexOf(TreeNode node);

[C++] public: int IndexOf(TreeNode* node);

[VB] Public Function IndexOf(ByVal node As TreeNode) As Integer

[JScript] public function IndexOf(node : TreeNode) : int;

12
13
14
15
16
17
18
19
20
21
22
23
24
25
Description

Returns the index of the specified tree node in the collection.

Return Value: The zero-based index of the item found in the tree node
collection; otherwise, -1. The **System.Windows.Forms.TreeNode** to locate in
the collection.

p) *Insert*

[C#] public virtual void Insert(int index, TreeNode node);

[C++] public: virtual void Insert(int index, TreeNode* node);

[VB] Overridable Public Sub Insert(ByVal index As Integer, ByVal node As
TreeNode)

[JScript] public function Insert(index : int, node : TreeNode);

Description

Inserts an existing tree node in the tree node collection at the specified location.

If the **System.Windows.Forms.TreeView.Sorted** property is set to **true** , the *index* parameter value is ignored. The **System.Windows.Forms.TreeNode** is inserted into the tree view and the **System.Windows.Forms.TreeView** resorted. The indexed location within the collection to insert the tree node. The **System.Windows.Forms.TreeNode** to insert into the collection.

q) *Remove*

```
[C#]          public          void          Remove(TreeNode          node);
[C++]          public:          void          Remove(TreeNode*          node);
[VB]    Public    Sub    Remove(ByVal    node    As    TreeNode)
[JScript]    public    function    Remove(node    :    TreeNode);
```

Description

Removes the specified tree node from the tree node collection.

When a **System.Windows.Forms.TreeNode** is removed from the tree node collection, all subsequent tree nodes are moved up one position in the collection. The **System.Windows.Forms.TreeNode** to remove.

r) *RemoveAt*

```
[C#]          public          virtual    void          RemoveAt(int          index);
[C++]          public:          virtual    void          RemoveAt(int          index);
[VB]    Overridable    Public    Sub    RemoveAt(ByVal    index    As    Integer)
[JScript]    public    function    RemoveAt(index    :    int);
```

Description

Removes a tree node from the tree node collection at a specified index.

When a **System.Windows.Forms.TreeNode** is removed from the tree node collection, all subsequent tree nodes are moved up one position in the collection. The index of the **System.Windows.Forms.TreeNode** to remove.

s) ***IList.Add***

[C#] int IList.Add(object node);

[C++] int IList::Add(Object* node);

[VB] Function Add(ByVal node As Object) As Integer Implements IList.Add

[JScript] function IList.Add(node : Object) : int;

t) ***IList.Contains***

[C#] bool IList.Contains(object node);

[C++] bool IList::Contains(Object* node);

[VB] Function Contains(ByVal node As Object) As Boolean Implements
IList.Contains

[JScript] function IList.Contains(node : Object) : Boolean;

u) ***IList.IndexOf***

[C#] int IList.IndexOf(object node);

[C++] int IList::IndexOf(Object* node);

[VB] Function IndexOf(ByVal node As Object) As Integer Implements
IList.IndexOf

[JScript] function IList.IndexOf(node : Object) : int;

v) ***IList.Insert***

[C#] void IList.Insert(int index, object node);

[C++] void IList::Insert(int index, Object* node);

[VB] Sub Insert(ByVal index As Integer, ByVal node As Object) Implements

1 IList.Insert

2 [JScript] function IList.Insert(index : int, node : Object);

3 w) *IList.Remove*

4
5 [C#] void IList.Remove(object node);

6 [C++] void IList::Remove(Object* node);

7 [VB] Sub Remove(ByVal node As Object) Implements IList.Remove

8 [JScript] function IList.Remove(node : Object);

9 TreeNodeConverter class (System.Windows.Forms)

10 a) *ToString*

11
12
13 *Description*

14 Provides a type converter to convert **System.Windows.Forms.TreeNode**
15 objects to and from various other representations.

16 For more information about type converters, see the
17 **System.ComponentModel.TypeConverter** base class and .

18 b) *TreeNodeConverter*

19 *Example Syntax:*

20 c) *ToString*

21 [C#] public TreeNodeConverter();

22 [C++] public: TreeNodeConverter();

23 [VB] Public Sub New()

24 [JScript] public function TreeNodeConverter();

d) *CanConvertTo*

```
[C#] public override bool CanConvertTo(ITypeDescriptorContext context, Type destinationType);
```

```
[C++] public: bool CanConvertTo(ITypeDescriptorContext* context, Type* destinationType);
```

```
[VB] Overrides Public Function CanConvertTo(ByVal context As ITypeDescriptorContext, ByVal destinationType As Type) As Boolean
```

```
[JScript] public override function CanConvertTo(context : ITypeDescriptorContext, destinationType : Type) : Boolean;
```

Description

Gets a value indicating whether this converter can convert an object to the given destination type using the context.

Return Value: **true** if this converter can perform the conversion; otherwise, **false**.

The *context* parameter can be used to extract additional information about the environment this converter is being invoked from. This can be **null**, so always check. Also, properties on the context object can return **null**. An **System.ComponentModel.ITypeDescriptorContext** that provides a format context. A **System.Type** that represents the type you wish to convert to.

e) *ConvertTo*

```
[C#] public override object ConvertTo(ITypeDescriptorContext context, CultureInfo culture, object value, Type destinationType);
```

```
[C++] public: Object* ConvertTo(ITypeDescriptorContext* context, CultureInfo* culture, Object* value, Type* destinationType);
```

```
[VB] Overrides Public Function ConvertTo(ByVal context As
```

```

1 ITypeDescriptorContext, ByVal culture As CultureInfo, ByVal value As Object,
2 ByVal destinationType As Type) As Object
3 [JScript] public override function ConvertTo(context : ITypeDescriptorContext,
4 culture : CultureInfo, value : Object, destinationType : Type) : Object;
5

```

Description

Converts the given object to another type. The most common types to convert are to and from a string object. The default implementation will make a call to ToString on the object if the object is valid and if the destination type is string. If this cannot convert to the destination type, this will throw a NotSupportedException.

Return Value: The converted object. A formatter context. This object can be used to extract additional information about the environment this converter is being invoked from. This may be null, so you should always check. Also, properties on the context object may also return null. An optional culture info. If not supplied the current culture is assumed. The object to convert. The type to convert the object to.

TreeView class (System.Windows.Forms)

a) ToString

Description

Displays a hierarchical collection of labeled items, each represented by a **System.Windows.Forms.TreeNode**.

The **System.Windows.Forms.TreeView.Nodes** collection holds all the **System.Windows.Forms.TreeNode** objects that are assigned to the **System.Windows.Forms.TreeView** control. The tree nodes in this collection are referred to as the root tree nodes. Any tree node that is subsequently added to a root tree node is referred to as a child node. Because each **System.Windows.Forms.TreeNode** can contain a collection of other **System.Windows.Forms.TreeNode** objects, you might find it difficult to determine your location in the tree structure when you iterate through the collection. You can parse the **System.Windows.Forms.TreeNode.FullPath** string by using the **System.Windows.Forms.TreeView.PathSeparator** string

value to determine where a **System.Windows.Forms.TreeNode** label begins and ends.

b) TreeView

Example Syntax:

c) ToString

[C#]	public	TreeView();
[C++]	public:	TreeView();
[VB]	Public	Sub New()
[JScript]	public	function TreeView();

Description

Initializes a new instance of the **System.Windows.Forms.TreeView** class.

- d) *AccessibilityObject*
- e) *AccessibleDefaultActionDescription*
- f) *AccessibleDescription*
- g) *AccessibleName*
- h) *AccessibleRole*
- i) *AllowDrop*
- j) *Anchor*
- k) *BackColor*
- l) *ToString*

Description

The background color for this control. Specifying null for this parameter sets the control's background color to its parent's background color.

- m) *BackgroundImage*
- n) *ToString*

```
[C#]    public    override    Image    BackgroundImage    {get;    set;}
[C++]    public:    __property    virtual    Image*    get_BackgroundImage();public:
__property    virtual    void    set_BackgroundImage(Image*);
[VB]    Overrides    Public    Property    BackgroundImage    As    Image
[JScript]    public function get BackgroundImage() : Image;public function set
BackgroundImage(Image);
```

1
2 *Description*

3
4 o) *BindingContext*

5 p) *BorderStyle*

6 q) *ToString*
7
8

9 *Description*

10 Gets or sets the border style of the tree view control.

11 When the **System.Windows.Forms.TreeView.BorderStyle** property is set to
12 **System.Windows.Forms.BorderStyle.Fixed3D** , the
13 **System.Windows.Forms.TreeView** has a sunken, three-dimensional
14 appearance. To display a flat, thin border around the
15 **System.Windows.Forms.TreeView** control, set the
16 **System.Windows.Forms.BorderStyle** property to
17 **System.Windows.Forms.BorderStyle.FixedSingle** .
18
19
20
21
22
23
24
25

- r) *Bottom*
- s) *Bounds*
- t) *CanFocus*
- u) *CanSelect*
- v) *Capture*
- w) *CausesValidation*
- x) *CheckBoxes*
- y) *ToString*

Description

Gets or sets a value indicating whether check boxes are displayed next to the tree nodes in the tree view control.

A check box is displayed to the left of both the tree node label and tree node **System.Drawing.Image** , if any. Check boxes allow the user to select more than one tree node at a time.

1 **z) *ClientRectangle***

2 **aa) *ClientSize***

3 **bb) *CompanyName***

4 **cc) *Container***

5 **dd) *ContainsFocus***

6 **ee) *ContextMenu***

7 **ff) *Controls***

8 **gg) *Created***

9 **hh) *CreateParams***

10 **ii) *ToString***

11
12
13 *Description*

14 **jj) *Cursor***

15 **kk) *DataBindings***

16 **ll) *DefaultImeMode***

17 **mm) *DefaultSize***

18 **nn) *ToString***

19
20
21
22 *Description*

23 Deriving classes can override this to configure a default size for their control.
24 This is more efficient than setting the size in the control's constructor.
25

oo) *DesignMode*
 pp) *DisplayRectangle*
 qq) *Disposing*
 rr) *Dock*
 ss) *Enabled*
 tt) *Events*
 uu) *Focused*
 vv) *Font*
 ww) *FontHeight*
 xx) *ForeColor*
 yy) *ToString*

Description

The current foreground color for this control, which is the color the control uses to draw its text.

zz) *FullRowSelect*

aaa) *ToString*

[C#] public bool FullRowSelect {get; set;}

[C++] public: __property bool get_FullRowSelect();public: __property void
set_FullRowSelect(bool);

[VB] Public Property FullRowSelect As Boolean

[JScript] public function get FullRowSelect() : Boolean;public function set

FullRowSelect(Boolean);

Description

Gets or sets a value indicating whether the selection highlight spans the width of the tree view control.

When **System.Windows.Forms.TreeView.FullRowSelect** is **true** , a selection highlight spans the entire width of the tree view, display rather than the width of just the tree node label. The **System.Windows.Forms.TreeView.FullRowSelect** property is ignored if **System.Windows.Forms.TreeView.ShowLines** is set to **true** .

bbb) Handle

ccc) HasChildren

ddd) Height

eee) HideSelection

fff) ToString

Description

Gets or sets a value indicating whether the selected tree node remains highlighted even when the tree view has lost the focus.

ggg) HotTracking

hhh) ToString

[C#] public bool HotTracking {get; set;}

[C++] public: __property bool get_HotTracking();public: __property void
set_HotTracking(bool);

[VB] Public Property HotTracking As Boolean

[JScript] public function get HotTracking() : Boolean;public function set
HotTracking(Boolean);

Description

Gets or sets a value indicating whether a tree node label takes on the appearance of a hyperlink as the mouse pointer passes over it.

If the **System.Windows.Forms.TreeView.CheckBoxes** property is set to **true** , the **System.Windows.Forms.TreeView.HotTracking** property has no effect.

iii) *ImageIndex*

jjj) *ToString*

[C#] public int ImageIndex {get; set;}

[C++] public: __property int get_ImageIndex();public: __property void
set_ImageIndex(int);

[VB] Public Property ImageIndex As Integer

[JScript] public function get ImageIndex() : int;public function set
ImageIndex(int);

Description

Gets or sets the image-list index value of the default image that is displayed by the tree nodes.

The **System.Windows.Forms.TreeView.ImageIndex** value is the index of an **System.Drawing.Image** stored in the **System.Windows.Forms.ImageList** object assigned to the **System.Windows.Forms.TreeView.ImageList** property.

1 *kkk) ImageList*

2 *lll) ToString*

3
4 [C#] public ImageList ImageList {get; set;}

5 [C++] public: __property ImageList* get_ImageList();public: __property void
6 set_ImageList(ImageList*);

7 [VB] Public Property ImageList As ImageList

8 [JScript] public function get ImageList() : ImageList;public function set
9 ImageList(ImageList);

10
11 *Description*

12 Gets or sets the **System.Windows.Forms.ImageList** that contains the
13 **System.Drawing.Image** objects used by the tree nodes.

14 If the **System.Windows.Forms.ImageList** property value is anything other
15 than **null** , all the tree nodes display the first **System.Drawing.Image** stored
16 in the **System.Windows.Forms.ImageList** .

17 *mmm) ImeMode*

18 *nnn) Indent*

19 *ooo) ToString*

20
21 *Description*

22 Gets or sets the distance to indent each of the child tree node levels.
23
24
25

1 *ppp) InvokeRequired*

2 *qqq) IsAccessible*

3 *rrr) IsDisposed*

4 *sss) IsHandleCreated*

5 *ttt) ItemHeight*

6 *uuu) ToString*

7
8
9 *Description*

10 Gets or sets the height of each tree node in the tree view control.

11 *vvv) LabelEdit*

12 *www) ToString*

13
14 [C#] public bool LabelEdit {get; set;}

15 [C++] public: __property bool get_LabelEdit();public: __property void
16 set_LabelEdit(bool);

17 [VB] Public Property LabelEdit As Boolean

18 [JScript] public function get LabelEdit() : Boolean;public function set
19 LabelEdit(Boolean);

20
21 *Description*

22 Gets or sets a value indicating whether the label text of the tree nodes can be
23 edited.

24 The **System.Windows.Forms.TreeNode.BeginEdit** method works only if the
25 **System.Windows.Forms.TreeView.LabelEdit** property is **true** . If
 System.Windows.Forms.TreeView.LabelEdit is **false** when you attempt to

edit the label, an exception will be thrown and the tree node will not be put in to an editable state.

xxx) Left

yyy) Location

zzz) Name

aaaa) Nodes

bbbb) ToString

Description

Gets the collection of tree nodes that are assigned to the tree view control.

The **System.Windows.Forms.TreeView.Nodes** property holds a collection of **System.Windows.Forms.TreeNode** objects, each of which has a **System.Windows.Forms.TreeNode.Nodes** property that can contain its own **System.Windows.Forms.TreeNodeCollection** . This nesting of tree nodes can make it difficult to navigate a tree structure, but the **System.Windows.Forms.TreeNode.FullPath** property makes it easier to determine your location within the tree structure.

cccc) Parent

dddd) PathSeparator

eeee) ToString

Description

Gets or sets the delimiter string that the tree node path uses.

The tree node path consists of a set of tree node labels separated by the **System.Windows.Forms.TreeView.PathSeparator** delimiter strings. The labels range from the root tree node to the tree node that you want.

ffff) *ProductName*

gggg) *ProductVersion*

hhhh) *RecreatingHandle*

iiii) *Region*

jjjj) *RenderRightToLeft*

kkkk) *ResizeRedraw*

llll) *Right*

mmmm) *RightToLeft*

nnnn) *Scrollable*

oooo) *ToString*

Description

Gets or sets a value indicating whether the tree view control displays scroll bars when they are needed.

If this property is set to **true** , scroll bars are displayed on the **System.Windows.Forms.TreeView** when any **System.Windows.Forms.TreeNode** object is located outside the control's client region.

pppp) *SelectedImageIndex*

qqqq) *ToString*

[C#] public int SelectedImageIndex {get; set;}

[C++] public: __property int get_SelectedImageIndex();public: __property void
set_SelectedImageIndex(int);

[VB] Public Property SelectedImageIndex As Integer

1 [JScript] public function get SelectedImageIndex() : int;public function set
2 SelectedImageIndex(int);

3
4 *Description*

5 Gets or sets the image list index value of the image that is displayed when a tree
6 node is selected.

7 The **System.Windows.Forms.TreeView.SelectedImageIndex** value is the
8 index of an **System.Drawing.Image** stored in the
9 **System.Windows.Forms.ImageList** object assigned to the
10 **System.Windows.Forms.TreeView.ImageList** property.

11
12 *rrrr) SelectedNode*

13 *ssss) ToString*

14 [C#] public TreeNode SelectedNode {get; set;}

15 [C++] public: __property TreeNode* get_SelectedNode();public: __property void
16 set_SelectedNode(TreeNode*);

17 [VB] Public Property SelectedNode As TreeNode

18 [JScript] public function get SelectedNode() : TreeNode;public function set
19 SelectedNode(TreeNode);

20
21 *Description*

22 Gets or sets the tree node that is currently selected in the tree view control.

23 If no **System.Windows.Forms.TreeNode** is currently selected, the
24 **System.Windows.Forms.TreeView.SelectedNode** property is **null** .
25

1 *tttt) ShowFocusCues*

2 *uuuu) ShowKeyboardCues*

3 *vvvv) ShowLines*

4 *www)ToString*

5
6
7 *Description*

8 Gets or sets a value indicating whether lines are drawn between tree nodes in
9 the tree view control.

10 If **System.Windows.Forms.TreeView.ShowLines** is set to **true** , the
11 **System.Windows.Forms.TreeView.FullRowSelect** property is ignored.

12 *xxxx) ShowPlusMinus*

13 *yyyy) ToString*

14 [C#] public bool ShowPlusMinus {get; set;}

15 [C++] public: __property bool get_ShowPlusMinus();public: __property void
16 set_ShowPlusMinus(bool);

17 [VB] Public Property ShowPlusMinus As Boolean

18 [JScript] public function get ShowPlusMinus() : Boolean;public function set
19 ShowPlusMinus(Boolean);

20
21 *Description*

22 Gets or sets a value indicating whether plus-sign (+) and minus-sign (-) buttons
23 are displayed next to tree nodes that contain child tree nodes.

24 The plus-sign and minus-sign buttons appear next to the root tree nodes only if
25 the **System.Windows.Forms.TreeView.ShowRootLines** property is **true** . If
26 the plus-sign and minus-sign buttons are not displayed, no visual cue exists to
27 indicate that the tree node contains child tree nodes and is expandable. The user

then must double-click a tree node to determine whether it contains child tree nodes, to expand it, or to collapse it.

zzzz) ShowRootLines

aaaaa) ToString

[C#] public bool ShowRootLines {get; set;}

[C++] public: __property bool get_ShowRootLines();public: __property void
set_ShowRootLines(bool);

[VB] Public Property ShowRootLines As Boolean

[JScript] public function get ShowRootLines() : Boolean;public function set
ShowRootLines(Boolean);

Description

Gets or sets a value indicating whether lines are drawn between the tree nodes that are at the root of the tree view.

If the **System.Windows.Forms.TreeView.ShowRootLines** property is **false**, the plus-sign and minus-sign buttons will not appear next to the root tree nodes, and the **System.Windows.Forms.TreeView.ShowPlusMinus** property will have no effect.

bbbbbb) Site

cccc) Size

dddddd) Sorted

eeee) ToString

Description

1 Gets or sets a value indicating whether the tree nodes in the tree view are
sorted.

2 When **System.Windows.Forms.TreeView.Sorted** is set to **true** , the
3 **System.Windows.Forms.TreeNode** objects are sorted in alphabetical order.

4 *fffff) TabIndex*

5 *ggggg) TabStop*

6 *hhhhh) Tag*

7 *iiii) Text*

8 *jjjj) ToString*

9
10
11 *Description*

12
13 *kkkkk) Top*

14 *llll) TopLevelControl*

15 *mmmmm) TopNode*

16 *nnnnn) ToString*

17
18
19 *Description*

20 Gets the first fully-visible tree node in the tree view control.

21 Initially, the **System.Windows.Forms.TreeView.TopNode** returns the first
22 root tree node, which is located at the top of the
23 **System.Windows.Forms.TreeView** . However, if the user has scrolled the
24 contents, another tree node might be at the top.
25

ooooo) *Visible*

ppppp) *VisibleCount*

qqqqq) *ToString*

Description

Gets the number of tree nodes that can be fully visible in the tree view control.

The **System.Windows.Forms.TreeView.VisibleCount** value can be greater than the number of tree nodes in the tree view. The value is calculated by dividing the height of the client window by the height of a tree node item. The result is the total number of **System.Windows.Forms.TreeNode** objects that the **System.Windows.Forms.TreeView** is capable of displaying within its current dimensions.

rrrrr) *Width*

sssss) *WindowTarget*

ttttt) *ToString*

Description

Occurs after the tree node check box is checked.

For more information about handling events, see .

uuuuu) *ToString*

[C#]	public	event	TreeViewEventHandler	AfterCollapse;
[C++]	public:	__event	TreeViewEventHandler*	AfterCollapse;
[VB]	Public	Event	AfterCollapse	As TreeViewEventHandler

Description

Occurs after the tree node is collapsed.

For more information about handling events, see .

vvvvv) ToString

[C#] public event TreeViewEventHandler AfterExpand;

[C++] public: __event TreeViewEventHandler* AfterExpand;

[VB] Public Event AfterExpand As TreeViewEventHandler

Description

Occurs after the tree node is expanded.

For more information about handling events, see .

wwwww) ToString

[C#] public event NodeLabelEditEventHandler AfterLabelEdit;

[C++] public: __event NodeLabelEditEventHandler* AfterLabelEdit;

[VB] Public Event AfterLabelEdit As NodeLabelEditEventHandler

Description

Occurs after the tree node label text is edited.

For more information about handling events, see .

xxxxx) ToString

[C#] public event TreeViewEventHandler AfterSelect;

1 [C++] public: __event TreeViewEventHandler* AfterSelect;

2 [VB] Public Event AfterSelect As TreeViewEventHandler

3
4 *Description*

5 Occurs after the tree node is selected.

6 For more information about handling events, see .

7 *yyyyy) ToString*

8
9
10 *Description*

11 Occurs before the tree node check box is checked.

12 For more information about handling events, see .

13 *zzzzz) ToString*

14
15 [C#] public event TreeViewCancelEventHandler BeforeCollapse;

16 [C++] public: __event TreeViewCancelEventHandler* BeforeCollapse;

17 [VB] Public Event BeforeCollapse As TreeViewCancelEventHandler

18
19 *Description*

20 Occurs before the tree node is collapsed.

21 For more information about handling events, see .

22 *aaaaaa)ToString*

23
24 [C#] public event TreeViewCancelEventHandler BeforeExpand;

25 [C++] public: __event TreeViewCancelEventHandler* BeforeExpand;

1 [VB] Public Event BeforeExpand As TreeViewCancelEventHandler

2
3 *Description*

4 Occurs before the tree node is expanded.

5 For more information about handling events, see .

6 *bbbbbb)ToString*

7
8 [C#] public event NodeLabelEditEventHandler BeforeLabelEdit;

9 [C++] public: __event NodeLabelEditEventHandler* BeforeLabelEdit;

10 [VB] Public Event BeforeLabelEdit As NodeLabelEditEventHandler

11
12 *Description*

13 Occurs before the tree node label text is edited.

14 For more information about handling events, see .

15 *cccccc)ToString*

16
17 [C#] public event TreeViewCancelEventHandler BeforeSelect;

18 [C++] public: __event TreeViewCancelEventHandler* BeforeSelect;

19 [VB] Public Event BeforeSelect As TreeViewCancelEventHandler

20
21 *Description*

22 Occurs before the tree node is selected.

23 For more information about handling events, see .

dddddd)ToString

Description

Occurs when an item is dragged into the tree view control.

For more information about handling events, see .

eeeeee)BeginUpdate

[C#]	public	void	BeginUpdate();
[C++]	public:	void	BeginUpdate();
[VB]	Public	Sub	BeginUpdate()
[JScript]	public	function	BeginUpdate();

Description

Disables any redrawing of the tree view.

To maintain performance while items are added one at a time to the **System.Windows.Forms.TreeView**, call the **System.Windows.Forms.TreeView.BeginUpdate** method. The **System.Windows.Forms.TreeView.BeginUpdate** method prevents the control from painting until the **System.Windows.Forms.TreeView.EndUpdate** method is called.

ffffff) CollapseAll

[C#]	public	void	CollapseAll();
[C++]	public:	void	CollapseAll();
[VB]	Public	Sub	CollapseAll()
[JScript]	public	function	CollapseAll();

Description

Collapses all the tree nodes.

The **System.Windows.Forms.TreeView.CollapseAll** method collapses all the **System.Windows.Forms.TreeNode** objects, including all the child tree nodes, that are in the **System.Windows.Forms.TreeView** control.

gggggg)CreateHandle

[C#]	protected	override	void	CreateHandle();
[C++]	protected:		void	CreateHandle();
[VB]	Overrides	Protected	Sub	CreateHandle()
[JScript]	protected	override	function	CreateHandle();

Description

hhhhh)Dispose

[C#]	protected	override	void	Dispose(bool disposing);
[C++]	protected:		void	Dispose(bool disposing);
[VB]	Overrides	Protected	Sub	Dispose(ByVal disposing As Boolean)
[JScript]	protected	override	function	Dispose(disposing : Boolean);

Description

iiii) EndUpdate

[C#]	public	void	EndUpdate();
------	--------	------	--------------

[C++]	public:	void	EndUpdate();
[VB]	Public	Sub	EndUpdate()
[JScript]	public	function	EndUpdate();

Description

Enables the redrawing of the tree view.

To maintain performance while items are added one at a time to the **System.Windows.Forms.TreeView**, call the **System.Windows.Forms.TreeView.BeginUpdate** method. The **System.Windows.Forms.TreeView.BeginUpdate** method prevents the control from painting until the **System.Windows.Forms.TreeView.EndUpdate** method is called.

jjjjj) ExpandAll

[C#]	public	void	ExpandAll();
[C++]	public:	void	ExpandAll();
[VB]	Public	Sub	ExpandAll()
[JScript]	public	function	ExpandAll();

Description

Expands all the tree nodes.

The **System.Windows.Forms.TreeView.ExpandAll** method expands all the **System.Windows.Forms.TreeNode** objects, including all the child tree nodes, that are in the **System.Windows.Forms.TreeView** control.

kkkkk) GetItemRenderStyles

[C#] protected OwnerDrawPropertyBag GetItemRenderStyles(TreeNode node, int state);

```

1 [C++] protected: OwnerDrawPropertyBag* GetItemRenderStyles(TreeNode*
2 node, int state);
3 [VB] Protected Function GetItemRenderStyles(ByVal node As TreeNode, ByVal
4 state As Integer) As OwnerDrawPropertyBag
5 [JScript] protected function GetItemRenderStyles(node : TreeNode, state : int) :
6 OwnerDrawPropertyBag;

```

Description

Generates colors for each item. This can be overridden to provide colors on a per state/per node basis, rather than using the ForeColor/BackColor/NodeFont properties on TreeNode.

Return Value: an OwnerDrawPropertyBag object containing the font and colors
Generates colors for each item. This can be overridden to provide colors on a per state/per node basis, rather than using the ForeColor/BackColor/NodeFont properties on TreeNode. the TreeNode currently being drawn the NMCUSTOMDRAW style bits for this node

IIIIII) GetNodeAt

```

16 [C#] public TreeNode GetNodeAt(Point pt);
17 [C++] public: TreeNode* GetNodeAt(Point pt);
18 [VB] Public Function GetNodeAt(ByVal pt As Point) As TreeNode
19 [JScript] public function GetNodeAt(pt : Point) : TreeNode; Retrieves the tree
20 node that is at the specified location.

```

Description

Retrieves the tree node that is at the specified point.

Return Value: The **System.Windows.Forms.TreeNode** at the specified point, in tree view coordinates.

You can pass the **System.Windows.Forms.MouseEventArgs.X** and **System.Windows.Forms.MouseEventArgs.Y** coordinates of the **System.Windows.Forms.Control.MouseDown** event as the **System.Drawing.Point.X** and **System.Drawing.Point.Y** values of a new **System.Drawing.Point** object. The **System.Drawing.Point** to evaluate and retrieve the node from.

mmmmmm)GetNodeAt

```
[C#]      public      TreeNode      GetNodeAt(int      x,      int      y);
[C++]     public:      TreeNode*      GetNodeAt(int      x,      int      y);
[VB] Public Function GetNodeAt(ByVal x As Integer, ByVal y As Integer) As
TreeNode
[JScrip] public function GetNodeAt(x : int, y : int) : TreeNode;
```

Description

Retrieves the tree node at the point with the specified coordinates.

Return Value: The **System.Windows.Forms.TreeNode** at the specified location, in tree view coordinates.

You can pass the **System.Windows.Forms.MouseEventArgs.X** and **System.Windows.Forms.MouseEventArgs.Y** coordinates of the **System.Windows.Forms.Control.MouseDown** event as the *x* and *y* parameters. The **System.Drawing.Point.X** position to evaluate and retrieve the node from. The **System.Drawing.Point.Y** position to evaluate and retrieve the node from.

nnnnnn)GetNodeCount

```
[C#]      public      int      GetNodeCount(bool      includeSubTrees);
[C++]     public:      int      GetNodeCount(bool      includeSubTrees);
[VB] Public Function GetNodeCount(ByVal includeSubTrees As Boolean) As
Integer
```

1 [JScript] public function GetNodeCount(includeSubTrees : Boolean) : int;

3 *Description*

4 Retrieves the number of tree nodes, optionally including those in all subtrees,
assigned to the tree view control.

5 *Return Value:* The number of tree nodes, optionally including those in all
subtrees, assigned to the tree view control.

6 If *includeSubTrees* is **true** , the result is the number of all the tree nodes in the
entire tree structure. **true** to count the **System.Windows.Forms.TreeNode**
7 items that the subtrees contain; otherwise, **false**.

8
9 *oooooo)IsInputKey*

10
11 [C#] protected override bool IsInputKey(Keys keyData);

12 [C++] protected: bool IsInputKey(Keys keyData);

13 [VB] Overrides Protected Function IsInputKey(ByVal keyData As Keys) As
14 Boolean

15 [JScript] protected override function IsInputKey(keyData : Keys) : Boolean;

16
17 *Description*

18 Overridden to handle RETURN key.

19 *pppppp)OnAfterCheck*

20
21 [C#] protected virtual void OnAfterCheck(TreeViewEventArgs e);

22 [C++] protected: virtual void OnAfterCheck(TreeViewEventArgs* e);

23 [VB] Overridable Protected Sub OnAfterCheck(ByVal e As TreeViewEventArgs)

24 [JScript] protected function OnAfterCheck(e : TreeViewEventArgs);

Description

Raises the **System.Windows.Forms.TreeView.AfterCheck** event.

Raising an event invokes the event handler through a delegate. For more information, see . A **System.Windows.Forms.TreeViewEventArgs** that contains the event data.

qqqqqq)OnAfterCollapse

```
[C#]    protected    virtual    void    OnAfterCollapse(TreeViewEventArgs e);
[C++]    protected:    virtual    void    OnAfterCollapse(TreeViewEventArgs* e);
[VB]    Overridable    Protected    Sub    OnAfterCollapse(ByVal e As
TreeViewEventArgs)
[JScript]    protected    function    OnAfterCollapse(e : TreeViewEventArgs);
```

Description

Raises the **System.Windows.Forms.TreeView.AfterCollapse** event.

Raising an event invokes the event handler through a delegate. For more information, see . A **System.Windows.Forms.TreeViewEventArgs** that contains the event data.

rrrrrr) OnAfterExpand

```
[C#]    protected    virtual    void    OnAfterExpand(TreeViewEventArgs e);
[C++]    protected:    virtual    void    OnAfterExpand(TreeViewEventArgs* e);
[VB]    Overridable    Protected    Sub    OnAfterExpand(ByVal e As
TreeViewEventArgs)
[JScript]    protected    function    OnAfterExpand(e : TreeViewEventArgs);
```

Description

Raises the **System.Windows.Forms.TreeView.AfterExpand** event.

Raising an event invokes the event handler through a delegate. For more information, see . A **System.Windows.Forms.TreeViewEventArgs** that contains the event data.

sssss) OnAfterLabelEdit

[C#] protected virtual void OnAfterLabelEdit(NodeLabelEditEventArgs e);

[C++] protected: virtual void OnAfterLabelEdit(NodeLabelEditEventArgs* e);

[VB] Overridable Protected Sub OnAfterLabelEdit(ByVal e As NodeLabelEditEventArgs)

[JScript] protected function OnAfterLabelEdit(e : NodeLabelEditEventArgs);

Description

Raises the **System.Windows.Forms.TreeView.AfterLabelEdit** event.

Raising an event invokes the event handler through a delegate. For more information, see . A **System.Windows.Forms.NodeLabelEditEventArgs** that contains the event data.

ttttt) OnAfterSelect

[C#] protected virtual void OnAfterSelect(TreeViewEventArgs e);

[C++] protected: virtual void OnAfterSelect(TreeViewEventArgs* e);

[VB] Overridable Protected Sub OnAfterSelect(ByVal e As TreeViewEventArgs)

[JScript] protected function OnAfterSelect(e : TreeViewEventArgs);

Description

Raises the **System.Windows.Forms.TreeView.AfterSelect** event.

Raising an event invokes the event handler through a delegate. For more information, see . A **System.Windows.Forms.TreeViewEventArgs** that contains the event data.

uuuuuu)OnBeforeCheck

[C#] protected virtual void OnBeforeCheck(TreeViewCancelEventArgs e);

[C++] protected: virtual void OnBeforeCheck(TreeViewCancelEventArgs* e);

[VB] Overridable Protected Sub OnBeforeCheck(ByVal e As TreeViewCancelEventArgs)

[JScript] protected function OnBeforeCheck(e : TreeViewCancelEventArgs);

Description

Raises the **System.Windows.Forms.TreeView.BeforeCheck** event.

Raising an event invokes the event handler through a delegate. For more information, see . A **System.Windows.Forms.TreeViewCancelEventArgs** that contains the event data.

vvvvvv)OnBeforeCollapse

[C#] protected virtual void OnBeforeCollapse(TreeViewCancelEventArgs e);

[C++] protected: virtual void OnBeforeCollapse(TreeViewCancelEventArgs* e);

[VB] Overridable Protected Sub OnBeforeCollapse(ByVal e As TreeViewCancelEventArgs)

[JScript] protected function OnBeforeCollapse(e : TreeViewCancelEventArgs);

Description

Raises the **System.Windows.Forms.TreeView.BeforeCollapse** event.

Raising an event invokes the event handler through a delegate. For more information, see . A **System.Windows.Forms.TreeViewCancelEventArgs** that contains the event data.

wwwwww)OnBeforeExpand

[C#] protected virtual void OnBeforeExpand(TreeViewCancelEventArgs e);

[C++] protected: virtual void OnBeforeExpand(TreeViewCancelEventArgs* e);

[VB] Overridable Protected Sub OnBeforeExpand(ByVal e As TreeViewCancelEventArgs)

[JScript] protected function OnBeforeExpand(e : TreeViewCancelEventArgs);

Description

Raises the **System.Windows.Forms.TreeView.BeforeExpand** event.

Raising an event invokes the event handler through a delegate. For more information, see . A **System.Windows.Forms.TreeViewCancelEventArgs** that contains the event data.

xxxxxx)OnBeforeLabelEdit

[C#] protected virtual void OnBeforeLabelEdit(NodeLabelEditEventArgs e);

[C++] protected: virtual void OnBeforeLabelEdit(NodeLabelEditEventArgs* e);

[VB] Overridable Protected Sub OnBeforeLabelEdit(ByVal e As NodeLabelEditEventArgs)

[JScript] protected function OnBeforeLabelEdit(e : NodeLabelEditEventArgs);

Description

Raises the **System.Windows.Forms.TreeView.BeforeLabelEdit** event.

Raising an event invokes the event handler through a delegate. For more information, see . A **System.Windows.Forms.NodeLabelEditEventArgs** that contains the event data.

yyyyyy)OnBeforeSelect

[C#] protected virtual void OnBeforeSelect(TreeViewCancelEventArgs e);

[C++] protected: virtual void OnBeforeSelect(TreeViewCancelEventArgs* e);

[VB] Overridable Protected Sub OnBeforeSelect(ByVal e As TreeViewCancelEventArgs)

[JScript] protected function OnBeforeSelect(e : TreeViewCancelEventArgs);

Description

Raises the **System.Windows.Forms.TreeView.BeforeSelect** event.

Raising an event invokes the event handler through a delegate. For more information, see . A **System.Windows.Forms.TreeViewCancelEventArgs** that contains the event data.

zzzzzz) OnHandleCreated

[C#] protected override void OnHandleCreated(EventArgs e);

[C++] protected: void OnHandleCreated(EventArgs* e);

[VB] Overrides Protected Sub OnHandleCreated(ByVal e As EventArgs)

[JScript] protected override function OnHandleCreated(e : EventArgs);

Description

aaaaaaa)OnHandleDestroyed

[C#] protected override void OnHandleDestroyed(EventArgs e);
[C++] protected: void OnHandleDestroyed(EventArgs* e);
[VB] Overrides Protected Sub OnHandleDestroyed(ByVal e As EventArgs)
[JScript] protected override function OnHandleDestroyed(e : EventArgs);

Description

bbbbbbb)OnItemDrag

[C#] protected virtual void OnItemDrag(ItemDragEventArgs e);
[C++] protected: virtual void OnItemDrag(ItemDragEventArgs* e);
[VB] Overridable Protected Sub OnItemDrag(ByVal e As ItemDragEventArgs)
[JScript] protected function OnItemDrag(e : ItemDragEventArgs);

Description

Raises the **System.Windows.Forms.TreeView.ItemDrag** event.

Raising an event invokes the event handler through a delegate. For more information, see . An **System.Windows.Forms.ItemDragEventArgs** that contains the event data.

ccccccc)OnKeyDown

[C#] protected override void OnKeyDown(KeyEventArgs e);

1 [C++] protected: void OnKeyDown(KeyEventArgs* e);

2 [VB] Overrides Protected Sub OnKeyDown(ByVal e As KeyEventArgs)

3 [JScript] protected override function OnKeyDown(e : KeyEventArgs);

4
5 *Description*

6 Handles the OnBeforeCheck / OnAfterCheck for keyboard clicks Handles the
7 OnBeforeCheck / OnAfterCheck for keyboard clicks

8
9
10 *ddddddd)OnKeyPress*

11 [C#] protected override void OnKeyPress(KeyPressEventArgs e);

12 [C++] protected: void OnKeyPress(KeyPressEventArgs* e);

13 [VB] Overrides Protected Sub OnKeyPress(ByVal e As KeyPressEventArgs)

14 [JScript] protected override function OnKeyPress(e : KeyPressEventArgs);

15
16 *Description*

17 Handles the OnBeforeCheck / OnAfterCheck for keyboard clicks Handles the
18 OnBeforeCheck / OnAfterCheck for keyboard clicks

19
20
21 *eeeeeee)OnKeyUp*

22 [C#] protected override void OnKeyUp(KeyEventArgs e);

23 [C++] protected: void OnKeyUp(KeyEventArgs* e);

24 [VB] Overrides Protected Sub OnKeyUp(ByVal e As KeyEventArgs)

25 [JScript] protected override function OnKeyUp(e : KeyEventArgs);

26
27 *Description*

Handles the OnBeforeCheck / OnAfterCheck for keyboard clicks Handles the
OnBeforeCheck / OnAfterCheck for keyboard clicks

ffffff) ToString

[C#]	public	override	string	ToString();
[C++]	public:		String*	ToString();
[VB]	Overrides	Public	Function	ToString() As String
[JScript]	public	override	function	ToString() : String;

Description

Returns a string representation for this control.

Return Value: String Returns a string representation for this control.

ggggggg) WndProc

[C#]	protected	override	void	WndProc(ref Message m);
[C++]	protected:		void	WndProc(Message* m);
[VB]	Overrides	Protected	Sub	WndProc(ByRef m As Message)
[JScript]	protected	override	function	WndProc(m : Message);

Description

TreeViewAction enumeration (System.Windows.Forms)

a) WndProc

Description

Specifies the action that raised a
System.Windows.Forms.TreeViewEventArgs event.

This enumeration is used by members such as the
System.Windows.Forms.TreeViewEventArgs.#ctor constructor.

b) WndProc

[C#]	public	const	TreeViewAction	ByKeyboard;
[C++]	public:	const	TreeViewAction	ByKeyboard;
[VB]	Public	Const	ByKeyboard	As TreeViewAction
[JScript]	public	var	ByKeyboard	: TreeViewAction;

Description

The event was caused by a keystroke.

c) WndProc

[C#]	public	const	TreeViewAction	ByMouse;
[C++]	public:	const	TreeViewAction	ByMouse;
[VB]	Public	Const	ByMouse	As TreeViewAction
[JScript]	public	var	ByMouse	: TreeViewAction;

Description

The event was caused by a mouse operation.

d) WndProc

[C#]	public	const	TreeViewAction	Collapse;
[C++]	public:	const	TreeViewAction	Collapse;

[VB]	Public	Const	Collapse	As	TreeViewAction
[JScript]	public	var	Collapse	:	TreeViewAction;

Description

The event was caused by the **System.Windows.Forms.TreeNode** collapsing.

e) WndProc

[C#]	public	const	TreeViewAction	Expand;
[C++]	public:	const	TreeViewAction	Expand;
[VB]	Public	Const	Expand	As TreeViewAction
[JScript]	public	var	Expand	: TreeViewAction;

Description

The event was caused by the **System.Windows.Forms.TreeNode** expanding.

f) WndProc

[C#]	public	const	TreeViewAction	Unknown;
[C++]	public:	const	TreeViewAction	Unknown;
[VB]	Public	Const	Unknown	As TreeViewAction
[JScript]	public	var	Unknown	: TreeViewAction;

Description

The action that caused the event is unknown.

TreeViewCancelEventArgs class (System.Windows.Forms)

a) *ToString*

Description

Provides data for the **System.Windows.Forms.TreeView.BeforeCheck** , **System.Windows.Forms.TreeView.BeforeCollapse** , **System.Windows.Forms.TreeView.BeforeExpand** , or **System.Windows.Forms.TreeView.BeforeSelect** events of a **System.Windows.Forms.TreeView** control.

For more information about handling events, see .

b) *TreeViewCancelEventArgs*

Example Syntax:

c) *ToString*

```
[C#] public TreeViewCancelEventArgs(TreeNode node, bool cancel,
TreeViewAction action);
```

```
[C++] public: TreeViewCancelEventArgs(TreeNode* node, bool cancel,
TreeViewAction action);
```

```
[VB] Public Sub New(ByVal node As TreeNode, ByVal cancel As Boolean,
ByVal action As TreeViewAction)
```

```
[JScript] public function TreeViewCancelEventArgs(node : TreeNode, cancel :
Boolean, action : TreeViewAction);
```

Description

Initializes a new instance of the **System.Windows.Forms.TreeViewCancelEventArgs** class with the specified

tree node, a value specifying whether the event is to be canceled, and the type of tree view action that raised the event. The **System.Windows.Forms.TreeNode** that the event is responding to. **true** to cancel the event; otherwise, **false**. The type of **System.Windows.Forms.TreeViewAction** that raised the event.

d) *Action*

e) *ToString*

[C#] public TreeViewAction Action {get;}

[C++] public: __property TreeViewAction get_Action();

[VB] Public ReadOnly Property Action As TreeViewAction

[JScript] public function get Action() : TreeViewAction;

Description

Gets the type of **System.Windows.Forms.TreeViewAction** that raised the event.

f) *Cancel*

g) *Node*

h) *ToString*

Description

Gets the tree node to be checked, expanded, collapsed, or selected.

TreeViewCancelEventHandler delegate (System.Windows.Forms)

a) *ToString*

Description

Represents the method that will handle the **System.Windows.Forms.TreeView.BeforeCheck** , **System.Windows.Forms.TreeView.BeforeCollapse** , **System.Windows.Forms.TreeView.BeforeExpand** , or **System.Windows.Forms.TreeView.BeforeSelect** event of a **System.Windows.Forms.TreeView** . The source of the event. A **System.Windows.Forms.TreeViewCancelEventArgs** that contains the event data.

When you create a **System.Windows.Forms.TreeViewCancelEventArgs** delegate, you identify the method that will handle the event. To associate the event with your event handler, add an instance of the delegate to the event. The event handler is called whenever the event occurs, unless you remove the delegate. For more information about event handler delegates, see .

TreeViewEventArgs class (System.Windows.Forms)

a) *ToString*

Description

Provides data for the **System.Windows.Forms.TreeView.AfterCheck** , **System.Windows.Forms.TreeView.AfterCollapse** , **System.Windows.Forms.TreeView.AfterExpand** , or **System.Windows.Forms.TreeView.AfterSelect** events of a **System.Windows.Forms.TreeView** control.

For more information about handling events, see .

b) *TreeViewEventArgs*

Example Syntax:

c) *ToString*

```
[C#]      public      TreeViewEventArgs(TreeNode      node);
[C++]      public:      TreeViewEventArgs(TreeNode*      node);
[VB]      Public      Sub      New(ByVal      node      As      TreeNode)
[JavaScript] public function TreeViewEventArgs(node : TreeNode); Initializes a new
instance of the System.Windows.Forms.TreeViewEventArgs class.
```

Description

Initializes a new instance of the **System.Windows.Forms.TreeViewEventArgs** class for the specified tree node. The **System.Windows.Forms.TreeNode** that the event is responding to.

d) *TreeViewEventArgs*

Example Syntax:

e) *ToString*

```
[C#] public TreeViewEventArgs(TreeNode node, TreeViewAction action);
[C++] public: TreeViewEventArgs(TreeNode* node, TreeViewAction action);
[VB] Public Sub New(ByVal node As TreeNode, ByVal action As
TreeViewAction)
[JavaScript] public function TreeViewEventArgs(node : `TreeNode, action :
TreeViewAction);
```

Description

Initializes a new instance of the **System.Windows.Forms.TreeViewEventArgs** class for the specified tree

node and with the specified type of action that raised the event. The **System.Windows.Forms.TreeNode** that the event is responding to. The type of **System.Windows.Forms.TreeViewAction** that raised the event.

f) Action

g) ToString

```
[C#]      public      TreeViewAction      Action      {get;}
[C++]      public:      __property      TreeViewAction      get_Action();
[VB]      Public      ReadOnly      Property      Action      As      TreeViewAction
[JScript]      public      function      get      Action()      :      TreeViewAction;
```

Description

Gets the type of action that raised the event.

h) Node

i) ToString

```
[C#]      public      TreeNode      Node      {get;}
[C++]      public:      __property      TreeNode*      get_Node();
[VB]      Public      ReadOnly      Property      Node      As      TreeNode
[JScript]      public      function      get      Node()      :      TreeNode;
```

Description

Gets the tree node that has been checked, expanded, collapsed, or selected.

TreeViewEventHandler delegate (System.Windows.Forms)

a) *ToString*

Description

Represents the method that will handle the **System.Windows.Forms.TreeView.AfterCheck** , **System.Windows.Forms.TreeView.AfterCollapse** , **System.Windows.Forms.TreeView.AfterExpand** , or **System.Windows.Forms.TreeView.AfterSelect** event of a **System.Windows.Forms.TreeView** . The source of the event. A **System.Windows.Forms.TreeViewEventArgs** that contains the event data.

When you create a **System.Windows.Forms.TreeViewEventArgs** delegate, you identify the method that will handle the event. To associate the event with your event handler, add an instance of the delegate to the event. The event handler is called whenever the event occurs, unless you remove the delegate. For more information about event handler delegates, see .

TreeViewImageIndexConverter class (System.Windows.Forms)

a) *ToString*

Description

Provides a type converter to convert data for an image index to and from one data type to another for use by the **System.Windows.Forms.TreeView** control.

For more information about type converters, see the **System.ComponentModel.TypeConverter** base class and .

b) *TreeViewImageIndexConverter*

Example Syntax:

c) *ToString*

```
[C#]          public          TreeViewImageIndexConverter();
[C++]          public:          TreeViewImageIndexConverter();
[VB]          Public          Sub          New()
[JScript] public function TreeViewImageIndexConverter();
```

d) *IncludeNoneAsStandardValue*

e) *ToString*

```
[C#]  protected  override  bool  IncludeNoneAsStandardValue  {get;}
[C++] protected: __property virtual bool get_IncludeNoneAsStandardValue();
[VB] Overrides Protected ReadOnly Property IncludeNoneAsStandardValue As
Boolean
[JScript] protected function get IncludeNoneAsStandardValue() : Boolean;
```

Description

Gets a value indicating whether a **none** or **null** value is valid in the **System.ComponentModel.TypeConverter.StandardValuesCollection** collection.

As implemented in this class, this property always returns **false** .

AxHost.TypeLibraryTimeStampAttribute class (System.Windows.Forms)

a) *ToString*

b) *AxHost.TypeLibraryTimeStampAttribute*

Example Syntax:

c) *ToString*

d) *TypeId*

e) *Value*

f) *ToString*

UICues enumeration (System.Windows.Forms)

a) *ToString*

Description

Specifies the state of the user interface.

This enumeration is used by members such as the **System.Windows.Forms.UICuesEventArgs.#ctor** constructor.

b) *ToString*

[C#]	public	const	UICues	Changed;
[C++]	public:	const	UICues	Changed;
[VB]	Public	Const	Changed	As UICues
[JScript]	public	var	Changed	: UICues;

Description

The state of the focus cues and keyboard cues has changed.

c) *ToString*

[C#]	public	const	UICues	ChangeFocus;
[C++]	public:	const	UICues	ChangeFocus;

[VB]	Public	Const	ChangeFocus	As	UICues
[JScript]	public	var	ChangeFocus	:	UICues;

Description

The state of the focus cues has changed.

d) ToString

[C#]	public	const	UICues	ChangeKeyboard;
[C++]	public:	const	UICues	ChangeKeyboard;
[VB]	Public	Const	ChangeKeyboard	As UICues
[JScript]	public	var	ChangeKeyboard	: UICues;

Description

The state of the keyboard cues has changed.

e) ToString

[C#]	public	const	UICues	None;
[C++]	public:	const	UICues	None;
[VB]	Public	Const	None	As UICues
[JScript]	public	var	None	: UICues;

Description

No change was made.

f) ToString

[C#]	public	const	UICues	ShowFocus;
[C++]	public:	const	UICues	ShowFocus;
[VB]	Public	Const	ShowFocus	As UICues
[JScript]	public	var	ShowFocus	: UICues;

Description

Focus rectangles are displayed after the change.

g) ToString

[C#]	public	const	UICues	ShowKeyboard;
[C++]	public:	const	UICues	ShowKeyboard;
[VB]	Public	Const	ShowKeyboard	As UICues
[JScript]	public	var	ShowKeyboard	: UICues;

Description

Keyboard cues are underlined after the change.

h) ToString

[C#]	public	const	UICues	Shown;
[C++]	public:	const	UICues	Shown;
[VB]	Public	Const	Shown	As UICues
[JScript]	public	var	Shown	: UICues;

1
2 *Description*

3 Focus rectangles are displayed and keyboard cues are underlined after the
4 change.

5 UICuesEventArgs class (System.Windows.Forms)

6 a) *ToString*

7
8
9 *Description*

10 Provides data for the **System.Windows.Forms.Control.ChangeUICues**
11 event.

12 A **System.Windows.Forms.UICuesEventArgs** specifies which user interface
13 feature changed and its new value.

14 b) *UICuesEventArgs*

15 *Example Syntax:*

16 c) *ToString*

17 [C#] public UICuesEventArgs(UICues uicues);
18 [C++] public: UICuesEventArgs(UICues uicues);
19 [VB] Public Sub New(ByVal uicues As UICues)
20 [JScript] public function UICuesEventArgs(uicues : UICues);

21
22 *Description*

23 Initializes a new instance of the **System.Windows.Forms.UICuesEventArgs**
24 class with the specified **System.Windows.Forms.UICues** . A bitwise
25 combination of the **System.Windows.Forms.UICues** values.

d) *Changed*

e) *ToString*

```
[C#]      public      UICues      Changed      {get;}
[C++]      public:      __property      UICues      get_Changed();
[VB]      Public      ReadOnly      Property      Changed      As      UICues
[JScript]      public      function      get      Changed()      :      UICues;
```

Description

Gets the bitwise combination of the **System.Windows.Forms.UICues** values.

f) *ChangeFocus*

g) *ToString*

```
[C#]      public      bool      ChangeFocus      {get;}
[C++]      public:      __property      bool      get_ChangeFocus();
[VB]      Public      ReadOnly      Property      ChangeFocus      As      Boolean
[JScript]      public      function      get      ChangeFocus()      :      Boolean;
```

Description

Gets a value indicating whether the state of the focus cues has changed.

h) *ChangeKeyboard*

i) *ToString*

```
[C#]      public      bool      ChangeKeyboard      {get;}
[C++]      public:      __property      bool      get_ChangeKeyboard();
```

```

1 [VB] Public ReadOnly Property ChangeKeyboard As Boolean
2 [JScript] public function get ChangeKeyboard() : Boolean;

```

Description

Gets a value indicating whether the state of the keyboard cues has changed.

j) ShowFocus

k) ToString

```

9 [C#] public bool ShowFocus {get;}
10 [C++] public: __property bool get_ShowFocus();
11 [VB] Public ReadOnly Property ShowFocus As Boolean
12 [JScript] public function get ShowFocus() : Boolean;

```

Description

Gets a value indicating whether focus rectangles are shown after the change.

l) ShowKeyboard

m) ToString

```

19 [C#] public bool ShowKeyboard {get;}
20 [C++] public: __property bool get_ShowKeyboard();
21 [VB] Public ReadOnly Property ShowKeyboard As Boolean
22 [JScript] public function get ShowKeyboard() : Boolean;

```

Description

Gets a value indicating whether keyboard cues are underlined after the change.

1 **UICuesEventHandler** delegate (System.Windows.Forms)

2 *a) ToString*

3
4
5 *Description*

6 Represents a method that will handle the
7 **System.Windows.Forms.Control.ChangeUICues** event of a
8 **System.Windows.Forms.Control** . The source of the event. A
9 **System.Windows.Forms.UICuesEventArgs** that contains the event data.

10 When you create a **System.Windows.Forms.UICuesEventHandler** delegate,
11 you identify the method that will handle the event. To associate the event with
12 your event handler, add an instance of the delegate to the event. The event
13 handler is called whenever the event occurs, unless you remove the delegate.
14 For more information about handling events with delegates, see .

15
16 **UpDownBase** class (System.Windows.Forms)

17 *a) ToString*

18
19 *Description*

20 Implements the basic functionality required by an up-down control.

21 The up-down control consists of a text box and a small vertical scroll bar,
22 commonly referred to as a spinner control. The
23 **System.Windows.Forms.UpDownBase** class links the two controls and
24 allows the user to change the display in the text box by clicking the up or down
25 buttons or by entering the appropriate type of value directly into the text box.
26 Use the up-down control in cases where you want to limit the list of values a
27 user may select, similar to a list box or combo box. Depending upon the type of
28 list you wish to display, the advantage to using an up-down control is that it
29 allows you to quickly set a range of valid values, rather than adding items one at
30 a time. Implementing an up-down control requires less data validation than a
31 text box, as you may limit the data type when you derive a class from
32 **System.Windows.Forms.UpDownBase** . An example of this is the
33 **System.Windows.Forms.NumericUpDown** class, which limits the values to
34 the numeric type and uses a

System.Windows.Forms.NumericUpDown.Minimum and **System.Windows.Forms.NumericUpDown.Maximum** property to validate the data.

b) UpDownBase

Example Syntax:

c) ToString

[C#]	public	UpDownBase();
[C++]	public:	UpDownBase();
[VB]	Public	Sub New()
[JScript]	public	function UpDownBase();

Description

Initializes a new instance of the **System.Windows.Forms.UpDownBase** class.

- d) *AccessibilityObject*
- e) *AccessibleDefaultActionDescription*
- f) *AccessibleDescription*
- g) *AccessibleName*
- h) *AccessibleRole*
- i) *ActiveControl*
- j) *AllowDrop*
- k) *Anchor*
- l) *AutoScroll*
- m) *ToString*

Description

- n) *AutoScrollMargin*
- o) *ToString*

```
[C#]      public      new      Size      AutoScrollMargin      {get;      set;}

[C++] public: __property Size get_AutoScrollMargin();public: __property void
set_AutoScrollMargin(Size);

[VB]      Public      Property      AutoScrollMargin      As      Size

[JScript] public function get AutoScrollMargin() : Size;public function set
AutoScrollMargin(Size);
```

Description

1 *p) AutoScrollMinSize*

2 *q) ToString*

3
4 [C#] public new Size AutoScrollMinSize {get; set;}

5 [C++] public: __property Size get_AutoScrollMinSize();public: __property void
6 set_AutoScrollMinSize(Size);

7 [VB] Public Property AutoScrollMinSize As Size

8 [JScript] public function get AutoScrollMinSize() : Size;public function set
9 AutoScrollMinSize(Size);

10
11 *Description*

12 *r) AutoScrollPosition*

13 *s) BackColor*

14 *t) ToString*

15
16
17 *Description*

18 Gets or sets the background color for the text box portion of the up-down
19 control.

20 *u) BackgroundImage*

21 *v) ToString*

22
23 [C#] public override Image BackgroundImage {get; set;}

24 [C++] public: __property virtual Image* get_BackgroundImage();public:
25 __property virtual void set_BackgroundImage(Image*);

1 [VB] Overrides Public Property BackgroundImage As Image
2 [JScript] public function get BackgroundImage() : Image;public function set
3 BackgroundImage(Image);
4

5 *Description*

6 w) *BindingContext*

7 x) *BorderStyle*

8 y) *ToString*
9

10
11 *Description*

12 Gets or sets the border style for the up-down control.

13 You can use the **System.Windows.Forms.TextBoxBase.BorderStyle**
14 property to create borderless and flat controls in addition to the default three-
15 dimensional control.
16
17
18
19
20
21
22
23
24
25

1 z) *Bottom*

2 aa) *Bounds*

3 bb) *CanFocus*

4 cc) *CanSelect*

5 dd) *Capture*

6 ee) *CausesValidation*

7 ff) *ChangingText*

8 gg) *ToString*

9
10
11 *Description*

12 Gets or sets a value indicating whether the text property is being changed
13 internally by its parent class.

14 The **System.Windows.Forms.UpDownBase.ChangingText** property acts as
15 a flag for the **System.Windows.Forms.UpDownBase** class. This property is
16 used by derived classes to indicate when the class is changing the current
17 **System.Windows.Forms.UpDownBase.Text** property internally. If this
18 property is set to **false** , the control assumes that the user is changing the
19 **System.Windows.Forms.UpDownBase.Text** property, and will set the
20 **System.Windows.Forms.UpDownBase.UserEdit** property to **true** .
21
22
23
24
25

1 **hh) ClientRectangle**

2 **ii) ClientSize**

3 **jj) CompanyName**

4 **kk) Container**

5 **ll) ContainsFocus**

6 **mm) ContextMenu**

7 **nn) ToString**

8
9
10 *Description*

11 The

12 The contextMenu associated with this control. The contextMenu will be shown
13 when the user right clicks the mouse on the control. We override this property
14 here in UpDownBase to set the contextmenu on the child edit control.

15 **oo) Controls**

16 **pp) Created**

17 **qq) CreateParams**

18 **rr) ToString**

19
20
21 *Description*

22 Returns the parameters needed to create the handle. Inheriting classes can
23 override this to provide extra functionality. They should not, however, forget to
24 call base.getCreateParams() first to get the struct filled up with the basic info.
25

- 1 *ss) Cursor*
- 2 *tt) DataBindings*
- 3 *uu) DefaultImeMode*
- 4 *vv) DefaultSize*
- 5 *ww) ToString*

8 *Description*

9 Deriving classes can override this to configure a default size for their control.
 10 This is more efficient than setting the size in the control's constructor.

- 11 *xx) DesignMode*
- 12 *yy) DisplayRectangle*
- 13 *zz) Disposing*
- 14 *aaa) Dock*
- 15 *bbb) DockPadding*
- 16 *ccc) ToString*

19 *Description*

ddd) *Enabled*

eee) *Events*

fff) *Focused*

ggg) *ToString*

Description

Returns true if this control has focus.

hhh) *Font*

iii) *FontHeight*

jjj) *ForeColor*

kkk) *ToString*

Description

Gets or sets the foreground color for the control.

1 **III) Handle**

2 **mmm) HasChildren**

3 **nnn) Height**

4 **ooo) HScroll**

5 **ppp) ImeMode**

6 **qqq) InterceptArrowKeys**

7 **rrr) ToString**

8

9

10 **Description**

11 Gets or sets a value indicating whether the user can use the UP ARROW and
12 DOWN ARROW keys to select values.

13 If **System.Windows.Forms.UpDownBase.InterceptArrowKeys** is set to
14 **true** and the up-down control has focus, the user may use the UP ARROW and
15 DOWN ARROW keys to select values.

1 *sss) InvokeRequired*

2 *ttt) IsAccessible*

3 *uuu) IsDisposed*

4 *vvv) IsHandleCreated*

5 *www) Left*

6 *xxx) Location*

7 *yyy) Name*

8 *zzz) Parent*

9 *aaaa) ParentForm*

10 *bbbb) PreferredHeight*

11 *cccc) ToString*

12
13
14 *Description*

15 Gets the height of the up-down control.

16 The **System.Windows.Forms.UpDownBase.PreferredHeight** property
17 value is based on the
18 **System.Windows.Forms.TextBoxBase.PreferredHeight** property of the
19 text box portion of the control and is adjusted for the style of border.

1 *dddd) ProductName*

2 *eeee) ProductVersion*

3 *ffff) ReadOnly*

4 *gggg) ToString*

5
6
7 *Description*

8 Gets or sets a value indicating whether the text may be changed by the use of
9 the up or down buttons only.

10 By setting the **System.Windows.FormsUpDownBase.ReadOnly** property to
11 **true** , you will eliminate the need for much validation of the
12 **System.Windows.FormsUpDownBase.Text** property. The user will be
13 restricted to the use of the up and down buttons to change the
14 **System.Windows.FormsUpDownBase.Text** values. It will only allow them
15 to select values you specify.

hhhh) RecreatingHandle
iiii) Region
jjjj) RenderRightToLeft
kkkk) ResizeRedraw
llll) Right
mmmm)RightToLeft
nnnn) ShowFocusCues
oooo) ShowKeyboardCues
pppp) Site
qqqq) Size
rrrr) TabIndex
ssss) TabStop
tttt) Tag
uuuu) Text
vvvv) ToString

Description

Gets or sets the text displayed in the up-down control.

If the **System.Windows.Forms.UpDownBase.Text** property is set while the **System.Windows.Forms.UpDownBase.UserEdit** property is set to **true** , the **System.Windows.Forms.UpDownBase.UpdateEditText** method will be called. If the **System.Windows.Forms.UpDownBase.Text** property is set while the **System.Windows.Forms.UpDownBase.UserEdit** property is set to **false** , **System.Windows.Forms.UpDownBase.ValidateEditText** will be called.

www) TextAlign

xxxx) ToString

```
[C#]    public    HorizontalAlignment    TextAlign    {get;    set;}
[C++]    public:    __property    HorizontalAlignment    get_TextAlign();public:
__property    void    set_TextAlign(HorizontalAlignment);
[VB]    Public    Property    TextAlign    As    HorizontalAlignment
[JScript] public function get TextAlign() : HorizontalAlignment;public function
set    TextAlign(HorizontalAlignment);
```

Description

Gets or sets the alignment of the text in the up-down control.

yyyy) Top

zzzz) TopLevelControl

aaaaa) UpDownAlign

bbbbbb) ToString

Description

Gets or sets the alignment of the up and down buttons on the up-down control.

cccc) UserEdit

dddddd) ToString

```
[C#]    protected    bool    UserEdit    {get;    set;}
```

[C++] protected: __property bool get_UserEdit();protected: __property void
set_UserEdit(bool);

[VB] Protected Property UserEdit As Boolean

[JScript] protected function get UserEdit() : Boolean;protected function set
UserEdit(Boolean);

Description

Gets or sets a value indicating whether a value has been entered by the user.

If the **System.Windows.Forms.UpDownBase.Text** property is set while the **System.Windows.Forms.UpDownBase.UserEdit** property is set to **true** , the **System.Windows.Forms.UpDownBase.UpdateEditText** method is called. If the **System.Windows.Forms.UpDownBase.Text** property is set while the **System.Windows.Forms.UpDownBase.UserEdit** property is set to **false** , **System.Windows.Forms.UpDownBase.ValidateEditText** is called.

eeeeee) Visible

ffffff) VScroll

ggggg) Width

hhhhh)WindowTarget

iiii) DownButton

[C#] public abstract void DownButton();

[C++] public: virtual void DownButton() = 0;

[VB] MustOverride Public Sub DownButton()

[JScript] public abstract function DownButton();

Description

When overridden in a derived class, handles the pressing of the down button on the up-down control.

When overriding this method in a derived class, be sure to call the **System.Windows.Forms.UpDownBase.UpdateEditText** method.

jjjjj) OnChanged

[C#] protected virtual void OnChanged(object source, EventArgs e);

[C++] protected: virtual void OnChanged(Object* source, EventArgs* e);

[VB] Overridable Protected Sub OnChanged(ByVal source As Object, ByVal e As EventArgs)

[JScript] protected function OnChanged(source : Object, e : EventArgs); When overridden in a derived class, raises the Changed event. event.

kkkkk) OnFontChanged

[C#] protected override void OnFontChanged(EventArgs e);

[C++] protected: void OnFontChanged(EventArgs* e);

[VB] Overrides Protected Sub OnFontChanged(ByVal e As EventArgs)

[JScript] protected override function OnFontChanged(e : EventArgs); Raises the FontChanged event.

lllll) OnHandleCreated

[C#] protected override void OnHandleCreated(EventArgs e);

[C++] protected: void OnHandleCreated(EventArgs* e);

[VB] Overrides Protected Sub OnHandleCreated(ByVal e As EventArgs)

[JScript] protected override function OnHandleCreated(e : EventArgs);

1
2 *Description*

3 Initialise the updown. Adds the edit and updown buttons.

4 *mmmmm)OnLayout*

5
6 [C#] protected override void OnLayout(LayoutEventArgs e);

7 [C++] protected: void OnLayout(LayoutEventArgs* e);

8 [VB] Overrides Protected Sub OnLayout(ByVal e As LayoutEventArgs)

9 [JScript] protected override function OnLayout(e : LayoutEventArgs);

10
11 *Description*

12 Handle the layout event. The size of the edit control, and the position of the
13 UpDown control must be modified.

14 *nnnnn)OnMouseWheel*

15 [C#] protected override void OnMouseWheel(MouseEventArgs e);

16 [C++] protected: void OnMouseWheel(MouseEventArgs* e);

17 [VB] Overrides Protected Sub OnMouseWheel(ByVal e As MouseEventArgs)

18 [JScript] protected override function OnMouseWheel(e : MouseEventArgs);

19
20
21 *Description*

22 Raises the **System.Windows.Forms.Control.MouseWheel** event.

23 If the up-down control has focus when this event occurs, the direction the mouse
24 wheel scrolled is determined and either the
25 **System.Windows.Forms.UpDownBase.UpButton** or
System.Windows.Forms.UpDownBase.DownButton method is called. A
System.Windows.Forms.MouseEventArgs that contains the event data.

ooooo) OnTextBoxKeyDown

[C#] protected virtual void OnTextBoxKeyDown(object source, EventArgs e);

[C++] protected: virtual void OnTextBoxKeyDown(Object* source, EventArgs* e);

[VB] Overridable Protected Sub OnTextBoxKeyDown(ByVal source As Object, ByVal e As EventArgs)

[JScript] protected function OnTextBoxKeyDown(source : Object, e : EventArgs);

Description

Raises the **System.Windows.Forms.Control.KeyDown** event.

If the **System.Windows.Forms.UpDownBase.InterceptArrowKeys** property is set to **true** and the key being pressed is the UP ARROW key, the **System.Windows.Forms.UpDownBase.UpButton** method is called.

Likewise, if the key being pressed is the DOWN ARROW key, the **System.Windows.Forms.UpDownBase.DownButton** method is called. The source of the event. A **System.Windows.Forms.KeyEventArgs** that contains the event data.

ppppp) OnTextBoxKeyPress

[C#] protected virtual void OnTextBoxKeyPress(object source, KeyPressEventArgs e);

[C++] protected: virtual void OnTextBoxKeyPress(Object* source, KeyPressEventArgs* e);

[VB] Overridable Protected Sub OnTextBoxKeyPress(ByVal source As Object, ByVal e As KeyPressEventArgs)

[JScript] protected function OnTextBoxKeyPress(source : Object, e :

KeyPressEventArgs);

Description

Raises the **System.Windows.Forms.Control.KeyPress** event.

Raising an event invokes the event handler through a delegate. For more information, see . The source of the event. A

System.Windows.Forms.KeyPressEventArgs that contains the event data.

qqqqq) OnTextBoxLostFocus

[C#] protected virtual void OnTextBoxLostFocus(object source, EventArgs e);

[C++] protected: virtual void OnTextBoxLostFocus(Object* source, EventArgs* e);

[VB] Overridable Protected Sub OnTextBoxLostFocus(ByVal source As Object, ByVal e As EventArgs)

[JScript] protected function OnTextBoxLostFocus(source : Object, e : EventArgs);

Description

Raises the **System.Windows.Forms.Control.LostFocus** event.

Validates the text in the text box portion of the control when the up-down control loses focus. The source of the event. An **System.EventArgs** that contains the event data.

rrrrr) OnTextBoxResize

[C#] protected virtual void OnTextBoxResize(object source, EventArgs e);

[C++] protected: virtual void OnTextBoxResize(Object* source, EventArgs* e);

[VB] Overridable Protected Sub OnTextBoxResize(ByVal source As Object,

1 ByVal e As EventArgs)

2 [JScript] protected function OnTextBoxResize(source : Object, e : EventArgs);

3
4 *Description*

5 Raises the **System.Windows.Forms.Control.Resize** event.

6 Adjusts the size of the up-down control when the text box portion of the control
7 is resized. The source of the event. An **System.EventArgs** that contains the
8 event data.

8 sssss) *OnTextBoxTextChanged*

9
10 [C#] protected virtual void OnTextBoxTextChanged(object source, EventArgs e);

11 [C++] protected: virtual void OnTextBoxTextChanged(Object* source,
12 EventArgs* e);

13 [VB] Overridable Protected Sub OnTextBoxTextChanged(ByVal source As
14 Object, ByVal e As EventArgs)

15 [JScript] protected function OnTextBoxTextChanged(source : Object, e :
16 EventArgs);

17
18 *Description*

19 Raises the **System.Windows.Forms.Control.TextChanged** event.

20 Raising an event invokes the event handler through a delegate. For more
21 information, see . The source of the event. An **System.EventArgs** that contains
22 the event data.

22 ttttt) *Select*

23
24 [C#] public void Select(int start, int length);

```

1 [C++]      public:      void      Select(int      start,      int      length);
2 [VB] Public Sub Select(ByVal start As Integer, ByVal length As Integer)
3 [JScript] public function Select(start : int, length : int); Selects a range of text in
4 the                up-down                control.

```

Description

Selects a range of text in the up-down control specifying the starting position and number of characters to select.

The

System.Windows.Forms.UpDownBase.Select(System.Int32, System.Int32) method can be used when the up-down control gets focus, or when the **System.Windows.Forms.UpDownBase.Text** property fails data validation. When adding the validation code for the **System.Windows.Forms.UpDownBase.ValidateEditText** method in a derived class, call the **System.Windows.Forms.UpDownBase.Select(System.Int32, System.Int32)** method when validation fails. The position of the first character to be selected. The total number of characters to be selected.

uuuuu)SetBoundsCore

```

16 [C#] protected override void SetBoundsCore(int x, int y, int width, int height,
17 BoundsSpecified                specified);
18 [C++] protected: void SetBoundsCore(int x, int y, int width, int height,
19 BoundsSpecified                specified);
20 [VB] Overrides Protected Sub SetBoundsCore(ByVal x As Integer, ByVal y As
21 Integer, ByVal width As Integer, ByVal height As Integer, ByVal specified As
22 BoundsSpecified)
23 [JScript] protected override function SetBoundsCore(x : int, y : int, width : int,
24 height      :      int,      specified      :      BoundsSpecified);
25

```


Description

Restricts the vertical size of the control Restricts the vertical size of the control

vvvvv) UpButton

[C#]	public	abstract	void	UpButton();
[C++]	public:	virtual	void	UpButton() = 0;
[VB]	MustOverride	Public	Sub	UpButton()
[JScript]	public	abstract	function	UpButton();

Description

When overridden in a derived class, handles the pressing of the up button on the up-down control.

When overriding this method in a derived class, be sure to call the **System.Windows.Forms.UpDownBase.UpdateEditText** method.

wwwww) UpdateEditText

[C#]	protected	abstract	void	UpdateEditText();
[C++]	protected:	virtual	void	UpdateEditText() = 0;
[VB]	MustOverride	Protected	Sub	UpdateEditText()
[JScript]	protected	abstract	function	UpdateEditText();

Description

When overridden in a derived class, updates the text displayed in the up-down control.

When overriding this method in a derived class, be sure to update the **System.Windows.Forms.UpDownBase.Text** property of the up-down control.

xxxxx) ValidateEditText

[C#]	protected	virtual	void	ValidateEditText();
[C++]	protected:	virtual	void	ValidateEditText();
[VB]	Overridable	Protected	Sub	ValidateEditText()
[JScript]	protected	function		ValidateEditText();

Description

When overridden in a derived class, validates the text displayed in the up-down control.

Examples of validation may include comparing the text entered to the data type set in your derived class, comparing it to a list of values, or verifying it to be within a range of values.

UpDownEventArgs class (System.Windows.Forms)

a) WndProc

Description

Provides data for the UpDownEvent

UpDownEventArgs This event is triggered when a button is pushed in the updown control. The event contains information on which button was pressed.

b) UpDownEventArgs

Example Syntax:

c) **WndProc**

```
[C#]          public          UpDownEventArgs(int          buttonPushed);
[C++]          public:          UpDownEventArgs(int          buttonPushed);
[VB]    Public    Sub    New(ByVal    buttonPushed    As    Integer)
[JScript]    public    function    UpDownEventArgs(buttonPushed    :    int);
```

Description

d) **ButtonID**

e) **WndProc**

```
[C#]          public          int          ButtonID          {get;}
[C++]          public:          __property          int          get_ButtonID();
[VB]    Public    ReadOnly    Property    ButtonID    As    Integer
[JScript]    public    function    get    ButtonID()    :    int;
```

Description

UpDownEventHandler delegate (System.Windows.Forms)

a) **ToString**

.

Description

UpDownEventHandler A delegate for an updown event handler.

UserControl class (System.Windows.Forms)

a) *ToString*

Description

Provides an empty control that can be used to create other controls.

By extending **System.Windows.Forms.ContainerControl** ,
System.Windows.Forms.UserControl inherits all the standard positioning
and mnemonic-handling code that is necessary in a user control.

b) *UserControl*

Example Syntax:

c) *ToString*

[C#]	public	UserControl();
[C++]	public:	UserControl();
[VB]	Public	Sub New()
[JScript]	public	function UserControl();

Description

Initializes a new instance of the **System.Windows.Forms.UserControl** class.

You do not typically create an instance of
System.Windows.Forms.UserControl . To create your own user control
class, inherit from the **System.Windows.Forms.UserControl** class.

- d) *AccessibilityObject*
- e) *AccessibleDefaultActionDescription*
- f) *AccessibleDescription*
- g) *AccessibleName*
- h) *AccessibleRole*
- i) *ActiveControl*
- j) *AllowDrop*
- k) *Anchor*
- l) *AutoScroll*
- m) *AutoScrollMargin*
- n) *AutoScrollMinSize*
- o) *AutoScrollPosition*
- p) *BackColor*
- q) *BackgroundImage*
- r) *BindingContext*
- s) *Bottom*
- t) *Bounds*
- u) *CanFocus*
- v) *CanSelect*
- w) *Capture*
- x) *CausesValidation*
- y) *ClientRectangle*
- z) *ClientSize*

1 **aa) *CompanyName***

2 **bb) *Container***

3 **cc) *ContainsFocus***

4 **dd) *ContextMenu***

5 **ee) *Controls***

6 **ff) *Created***

7 **gg) *CreateParams***

8 **hh) *Cursor***

9 **ii) *DataBindings***

10 **jj) *DefaultImeMode***

11 **kk) *DefaultSize***

12 **ll) *ToString***

13
14
15
16 ***Description***

17 The default size for this user control.
18
19
20
21
22
23
24
25

1 *jjj) Parent*
 2 *kkk) ParentForm*
 3 *lll) ProductName*
 4 *mmm) ProductVersion*
 5 *nnn) RecreatingHandle*
 6 *ooo) Region*
 7 *ppp) RenderRightToLeft*
 8 *qqq) ResizeRedraw*
 9 *rrr) Right*
 10 *sss) RightToLeft*
 11 *ttt) ShowFocusCues*
 12 *uuu) ShowKeyboardCues*
 13 *vvv) Site*
 14 *www) Size*
 15 *xxx) TabIndex*
 16 *yyy) TabStop*
 17 *zzz) Tag*
 18 *aaaa) Text*
 19 *bbbb) ToString*

Description

cccc) Top
dddd) TopLevelControl
eeee) Visible
ffff) VScroll
gggg) Width
hhhh) WindowTarget
iiii) ToString

Description

Occurs before the control becomes visible for the first time.

You can use this event to perform tasks such as allocating resources used by the control.

jjjj) OnCreateControl

[C#]	protected	override	void	OnCreateControl();
[C++]	protected:		void	OnCreateControl();
[VB]	Overrides	Protected	Sub	OnCreateControl()
[JScript]	protected	override	function	OnCreateControl();

Description

Raises the CreateControl event.

Raising an event invokes the event handler through a delegate. For more information, see .

kkkk) OnLoad

```
[C#]      protected      virtual      void      OnLoad(EventArgs      e);
[C++]      protected:      virtual      void      OnLoad(EventArgs*      e);
[VB]      Overridable      Protected      Sub      OnLoad(ByVal e As EventArgs)
[JScript]      protected      function      OnLoad(e      :      EventArgs);
```

Description

Raises the **System.Windows.Forms.UserControl.Load** event.

The **System.Windows.Forms.UserControl.Load** event occurs after the control is created, but before the control becomes visible for the first time. An **System.EventArgs** that contains the event data.

llll) OnMouseDown

```
[C#]      protected      override      void      OnMouseDown(MouseEventArgs      e);
[C++]      protected:      void      OnMouseDown(MouseEventArgs*      e);
[VB]      Overrides      Protected      Sub      OnMouseDown(ByVal e As MouseEventArgs)
[JScript]      protected      override      function      OnMouseDown(e : MouseEventArgs);
```

Description

mmmm)WndProc

```
[C#]      protected      override      void      WndProc(ref      Message      m);
[C++]      protected:      void      WndProc(Message*      m);
[VB]      Overrides      Protected      Sub      WndProc(ByRef m As Message)
```

1 [JScript] protected override function WndProc(m : Message);

3 *Description*

5 View enumeration (System.Windows.Forms)

6 a) *WndProc*

9 *Description*

10 Specifies how list items are displayed in a **System.Windows.Forms.ListView** control.

11 Use the members of this enumeration to set the value of the
12 **System.Windows.Forms.ListView.View** property of the
13 **System.Windows.Forms.ListView** control.

14 b) *WndProc*

16 [C#]	public	const	View	Details;
17 [C++]	public:	const	View	Details;
18 [VB]	Public	Const	Details	As View
19 [JScript]	public	var	Details	: View;

21 *Description*

22 Each item appears on a separate line with further information about each item
23 arranged in columns. The left most column contains a small icon and label, and
24 subsequent columns contain sub items as specified by the application. A column
25 displays a header which can display a caption for the column. The user can
resize each column at runtime.

c) *WndProc*

[C#]	public	const	View	LargeIcon;
[C++]	public:	const	View	LargeIcon;
[VB]	Public	Const	LargeIcon	As View
[JScript]	public	var	LargeIcon	: View;

Description

Each item appears as a full-sized icon with a label below it.

d) *WndProc*

[C#]	public	const	View	List;
[C++]	public:	const	View	List;
[VB]	Public	Const	List	As View
[JScript]	public	var	List	: View;

Description

Each item appears as a small icon with a label to its right. Items are arranged in columns with no column headers.

e) *WndProc*

[C#]	public	const	View	SmallIcon;
[C++]	public:	const	View	SmallIcon;
[VB]	Public	Const	SmallIcon	As View
[JScript]	public	var	SmallIcon	: View;

Description

Each item appears as a small icon with a label to its right.

VScrollBar class (System.Windows.Forms)

a) ToString

Description

Represents a standard Windows vertical scroll bar.

Most controls that need scroll bars already provide them and do not require this control. This is true of a multi-line **System.Windows.Forms.TextBox** control, a **System.Windows.Forms.ListBox** , and a **System.Windows.Forms.ComboBox** , for example.

b) VScrollBar

Example Syntax:

c) ToString

[C#]	public	VScrollBar();
[C++]	public:	VScrollBar();
[VB]	Public	Sub New()
[JScript]	public function	VScrollBar();

- d) *AccessibilityObject*
- e) *AccessibleDefaultActionDescription*
- f) *AccessibleDescription*
- g) *AccessibleName*
- h) *AccessibleRole*
- i) *AllowDrop*
- j) *Anchor*
- k) *BackColor*
- l) *BackgroundImage*
- m) *BindingContext*
- n) *Bottom*
- o) *Bounds*
- p) *CanFocus*
- q) *CanSelect*
- r) *Capture*
- s) *CausesValidation*
- t) *ClientRectangle*
- u) *ClientSize*
- v) *CompanyName*
- w) *Container*
- x) *ContainsFocus*
- y) *ContextMenu*
- z) *Controls*

1 *aa) Created*

2 *bb) CreateParams*

3 *cc) ToString*

4
5
6 *Description*

7 Returns the parameters needed to create the handle. Inheriting classes can
8 override this to provide extra functionality. They should not, however, forget to
9 call `base.CreateParams()` first to get the struct filled up with the basic info.

9 *dd) Cursor*

10 *ee) DataBindings*

11 *ff) DefaultImeMode*

12 *gg) DefaultSize*

13 *hh) ToString*

14
15
16 *Description*

17 Deriving classes can override this to configure a default size for their control.
18 This is more efficient than setting the size in the control's constructor.

19
20 EXEMPLARY COMPUTING SYSTEM AND ENVIRONMENT

21 Fig. 4 illustrates an example of a suitable computing environment 400
22 within which the programming framework 132 may be implemented (either fully
23 or partially). The computing environment 400 may be utilized in the computer
24 and network architectures described herein.

1 The exemplary computing environment 400 is only one example of a
2 computing environment and is not intended to suggest any limitation as to the
3 scope of use or functionality of the computer and network architectures. Neither
4 should the computing environment 400 be interpreted as having any dependency
5 or requirement relating to any one or combination of components illustrated in the
6 exemplary computing environment 400.

7 The framework 132 may be implemented with numerous other general
8 purpose or special purpose computing system environments or configurations.
9 Examples of well known computing systems, environments, and/or configurations
10 that may be suitable for use include, but are not limited to, personal computers,
11 server computers, multiprocessor systems, microprocessor-based systems, network
12 PCs, minicomputers, mainframe computers, distributed computing environments
13 that include any of the above systems or devices, and so on.

14 The framework 132 may be described in the general context of computer-
15 executable instructions, such as program modules, being executed by one or more
16 computers or other devices. Generally, program modules include routines,
17 programs, objects, components, data structures, etc. that perform particular tasks
18 or implement particular abstract data types. The framework 132 may also be
19 practiced in distributed computing environments where tasks are performed by
20 remote processing devices that are linked through a communications network. In
21 a distributed computing environment, program modules may be located in both
22 local and remote computer storage media including memory storage devices.

23 The computing environment 400 includes a general-purpose computing
24 device in the form of a computer 402. The components of computer 402 can
25 include, by are not limited to, one or more processors or processing units 404, a

1 system memory 406, and a system bus 408 that couples various system
2 components including the processor 404 to the system memory 406.

3 The system bus 408 represents one or more of several possible types of bus
4 structures, including a memory bus or memory controller, a peripheral bus, an
5 accelerated graphics port, and a processor or local bus using any of a variety of
6 bus architectures. By way of example, such architectures can include an Industry
7 Standard Architecture (ISA) bus, a Micro Channel Architecture (MCA) bus, an
8 Enhanced ISA (EISA) bus, a Video Electronics Standards Association (VESA)
9 local bus, and a Peripheral Component Interconnects (PCI) bus also known as a
10 Mezzanine bus.

11 Computer 402 typically includes a variety of computer readable media.
12 Such media can be any available media that is accessible by computer 402 and
13 includes both volatile and non-volatile media, removable and non-removable
14 media.

15 The system memory 406 includes computer readable media in the form of
16 volatile memory, such as random access memory (RAM) 410, and/or non-volatile
17 memory, such as read only memory (ROM) 412. A basic input/output system
18 (BIOS) 414, containing the basic routines that help to transfer information
19 between elements within computer 402, such as during start-up, is stored in ROM
20 412. RAM 410 typically contains data and/or program modules that are
21 immediately accessible to and/or presently operated on by the processing unit 404.

22 Computer 402 may also include other removable/non-removable,
23 volatile/non-volatile computer storage media. By way of example, Fig. 4
24 illustrates a hard disk drive 416 for reading from and writing to a non-removable,
25 non-volatile magnetic media (not shown), a magnetic disk drive 418 for reading

1 from and writing to a removable, non-volatile magnetic disk 420 (e.g., a “floppy
2 disk”), and an optical disk drive 422 for reading from and/or writing to a
3 removable, non-volatile optical disk 424 such as a CD-ROM, DVD-ROM, or other
4 optical media. The hard disk drive 416, magnetic disk drive 418, and optical disk
5 drive 422 are each connected to the system bus 408 by one or more data media
6 interfaces 426. Alternatively, the hard disk drive 416, magnetic disk drive 418,
7 and optical disk drive 422 can be connected to the system bus 408 by one or more
8 interfaces (not shown).

9 The disk drives and their associated computer-readable media provide non-
10 volatile storage of computer readable instructions, data structures, program
11 modules, and other data for computer 402. Although the example illustrates a
12 hard disk 416, a removable magnetic disk 420, and a removable optical disk 424,
13 it is to be appreciated that other types of computer readable media which can store
14 data that is accessible by a computer, such as magnetic cassettes or other magnetic
15 storage devices, flash memory cards, CD-ROM, digital versatile disks (DVD) or
16 other optical storage, random access memories (RAM), read only memories
17 (ROM), electrically erasable programmable read-only memory (EEPROM), and
18 the like, can also be utilized to implement the exemplary computing system and
19 environment.

20 Any number of program modules can be stored on the hard disk 416,
21 magnetic disk 420, optical disk 424, ROM 412, and/or RAM 410, including by
22 way of example, an operating system 426, one or more application programs 428,
23 other program modules 430, and program data 432. Each of the operating system
24 426, one or more application programs 428, other program modules 430, and
25

1 program data 432 (or some combination thereof) may include elements of the
2 programming framework 132.

3 A user can enter commands and information into computer 402 via input
4 devices such as a keyboard 434 and a pointing device 436 (e.g., a "mouse").
5 Other input devices 438 (not shown specifically) may include a microphone,
6 joystick, game pad, satellite dish, serial port, scanner, and/or the like. These and
7 other input devices are connected to the processing unit 404 via input/output
8 interfaces 440 that are coupled to the system bus 408, but may be connected by
9 other interface and bus structures, such as a parallel port, game port, or a universal
10 serial bus (USB).

11 A monitor 442 or other type of display device can also be connected to the
12 system bus 408 via an interface, such as a video adapter 444. In addition to the
13 monitor 442, other output peripheral devices can include components such as
14 speakers (not shown) and a printer 446 which can be connected to computer 402
15 via the input/output interfaces 440.

16 Computer 402 can operate in a networked environment using logical
17 connections to one or more remote computers, such as a remote computing device
18 448. By way of example, the remote computing device 448 can be a personal
19 computer, portable computer, a server, a router, a network computer, a peer device
20 or other common network node, and so on. The remote computing device 448 is
21 illustrated as a portable computer that can include many or all of the elements and
22 features described herein relative to computer 402.

23 Logical connections between computer 402 and the remote computer 448
24 are depicted as a local area network (LAN) 450 and a general wide area network
25

1 (WAN) 452. Such networking environments are commonplace in offices,
2 enterprise-wide computer networks, intranets, and the Internet.

3 When implemented in a LAN networking environment, the computer 402 is
4 connected to a local network 450 via a network interface or adapter 454. When
5 implemented in a WAN networking environment, the computer 402 typically
6 includes a modem 456 or other means for establishing communications over the
7 wide network 452. The modem 456, which can be internal or external to computer
8 402, can be connected to the system bus 408 via the input/output interfaces 440 or
9 other appropriate mechanisms. It is to be appreciated that the illustrated network
10 connections are exemplary and that other means of establishing communication
11 link(s) between the computers 402 and 448 can be employed.

12 In a networked environment, such as that illustrated with computing
13 environment 400, program modules depicted relative to the computer 402, or
14 portions thereof, may be stored in a remote memory storage device. By way of
15 example, remote application programs 458 reside on a memory device of remote
16 computer 448. For purposes of illustration, application programs and other
17 executable program components such as the operating system are illustrated herein
18 as discrete blocks, although it is recognized that such programs and components
19 reside at various times in different storage components of the computing device
20 402, and are executed by the data processor(s) of the computer.

21 An implementation of the framework 132, and particularly, the API 142 or
22 calls made to the API 142, may be stored on or transmitted across some form of
23 computer readable media. Computer readable media can be any available media
24 that can be accessed by a computer. By way of example, and not limitation,
25 computer readable media may comprise "computer storage media" and

1 “communications media.” “Computer storage media” include volatile and non-
2 volatile, removable and non-removable media implemented in any method or
3 technology for storage of information such as computer readable instructions, data
4 structures, program modules, or other data. Computer storage media includes, but
5 is not limited to, RAM, ROM, EEPROM, flash memory or other memory
6 technology, CD-ROM, digital versatile disks (DVD) or other optical storage,
7 magnetic cassettes, magnetic tape, magnetic disk storage or other magnetic storage
8 devices, or any other medium which can be used to store the desired information
9 and which can be accessed by a computer.

10 “Communication media” typically embodies computer readable
11 instructions, data structures, program modules, or other data in a modulated data
12 signal, such as carrier wave or other transport mechanism. Communication media
13 also includes any information delivery media. The term “modulated data signal”
14 means a signal that has one or more of its characteristics set or changed in such a
15 manner as to encode information in the signal. By way of example, and not
16 limitation, communication media includes wired media such as a wired network or
17 direct-wired connection, and wireless media such as acoustic, RF, infrared, and
18 other wireless media. Combinations of any of the above are also included within
19 the scope of computer readable media.

20 Alternatively, portions of the framework may be implemented in hardware
21 or a combination of hardware, software, and/or firmware. For example, one or
22 more application specific integrated circuits (ASICs) or programmable logic
23 devices (PLDs) could be designed or programmed to implement one or more
24 portions of the framework.

Conclusion

Although the invention has been described in language specific to structural features and/or methodological acts, it is to be understood that the invention defined in the appended claims is not necessarily limited to the specific features or acts described. Rather, the specific features and acts are disclosed as exemplary forms of implementing the claimed invention.